

# ROBUSTLY SOLVABLE CONSTRAINT SATISFACTION PROBLEMS\*

LIBOR BARTO<sup>†</sup> AND MARCIN KOZIK<sup>‡</sup>

**Abstract.** An algorithm for a constraint satisfaction problem is called robust if it outputs an assignment satisfying at least  $(1 - g(\varepsilon))$ -fraction of the constraints given a  $(1 - \varepsilon)$ -satisfiable instance, where  $g(\varepsilon) \rightarrow 0$  as  $\varepsilon \rightarrow 0$ . Guruswami and Zhou conjectured a characterization of constraint languages for which the corresponding constraint satisfaction problem admits an efficient robust algorithm. This paper confirms their conjecture.

**Key words.** constraint satisfaction problem, bounded width, approximation, robust satisfiability, universal algebra

**AMS subject classifications.** 68Q17, 68W20, 68W25, 68W40

**1. Introduction.** The constraint satisfaction problem (CSP) provides a common framework for many theoretical problems in computer science as well as for many applications. An instance of the CSP consists of variables and constraints imposed on them and the goal is to find (or decide whether it exists) an assignment of variables which is “best” for given constraints. In the decision problem for CSP we want to decide if there is an assignment satisfying all the constraints. In Max-CSP we wish to find an assignment satisfying maximal number of constraints. In the approximation version of Max-CSP we seek an assignment which is, in some sense, close to the optimal one. This paper deals with a special case of approximation: robust solvability of the CSP. Given an instance which is almost satisfiable (say  $(1 - \varepsilon)$ -fraction of the constraint can be satisfied), the goal is to efficiently find an almost satisfying assignment (which satisfies at least  $(1 - g(\varepsilon))$ -fraction of the constraints, where the error function  $g$  satisfies  $\lim_{\varepsilon \rightarrow 0} g(\varepsilon) = 0$ ).

Most of the computational problems connected to CSP are hard in general. Therefore, when developing algorithms, one usually restricts the set of allowed instances. Most often the instances are restricted in two ways: one restricts the way in which the variables are constrained (e.g. the shape of the hypergraph of constrained variables), or restricts the allowed constraint relations (defining *constraint language*). In this paper we use the second approach, i.e. all constraint relations must come from a fixed, finite set of relations on a domain.

Robust solvability for a fixed constraint language was first studied in a paper by Zwick [30]. The motivation behind this approach was that, in certain practical situations, instances might be close to satisfiable – for example, a small fraction of constraints might have been corrupted by noise. An algorithm that is able to satisfy, in such a case, most of the constraints could be useful.

Zwick [30] concentrated on Boolean CSPs. He designed a semidefinite programming (SDP) based algorithm which finds  $(1 - O(\varepsilon^{1/3}))$ -satisfying assignment for  $(1 - \varepsilon)$ -satisfiable instances of 2-SAT and linear programming (LP) based algorithm which

---

\*Parts of this work appeared in proceedings of STOC’12.

<sup>†</sup>Department of Algebra, Faculty of Mathematics and Physics, Charles University in Prague, Sokolovská 83, 18675 Praha 8, Czech Republic, [libor.barto@gmail.com](mailto:libor.barto@gmail.com). Research supported by the Grant Agency of the Czech Republic, grant 13-01832S;

<sup>‡</sup>Theoretical Computer Science Department, Faculty of Mathematics and Computer Science Jagiellonian University ul. Prof. St. Łojasiewicza 6, 30-348 Krakow, Poland, [Marcin.Kozik@uj.edu.pl](mailto:Marcin.Kozik@uj.edu.pl). Research partially supported by National Science Center grant no. DEC-2011/01/B/ST6/01006.

finds  $(1 - O(1/\log(1/\varepsilon)))$ -satisfying assignment for  $(1 - \varepsilon)$ -satisfiable instances of **Horn- $k$ -Sat** (the number  $k$  refers to the maximum numbers of variables in a Horn constraint). The quantitative dependence on  $\varepsilon$  was improved for **2-SAT** to  $(1 - O(\sqrt{\varepsilon}))$  in [10]. For **CUT**, a special case of **2-SAT**, the Goemans-Williamson algorithm [14] also achieves  $(1 - O(\sqrt{\varepsilon}))$ . The same dependence was proved more generally for **Unique-Games**( $q$ ) [9] (where  $q$  refers to the size of the domain), which improved  $(1 - O(\sqrt[q]{\varepsilon} \log^{1/2}(1/\varepsilon)))$  obtained in [21]. For **Horn-2-Sat** the exponential loss can be replaced by  $(1 - 3\varepsilon)$  [20] and even  $(1 - 2\varepsilon)$  [15]. These bounds for **Horn- $k$ -Sat** ( $k \geq 3$ ), **Horn-2-Sat**, **2-SAT**, and **Unique-Games**( $q$ ) are actually essentially optimal [21, 22, 15] assuming Khot’s Unique Games Conjecture [21].

On the negative side, if the decision problem for CSP is NP-complete for algebraic reasons (for precise definition see [8, 6]) then, given a satisfiable instance, it is NP-hard to find an assignment satisfying  $\alpha$ -fraction of the constraints for some constant  $\alpha < 1$  (see [20] for the Boolean case and [18] for the general case). In particular, these problems cannot admit an efficient robust algorithm unless  $P=NP$ . However, this is not the only obstacle for robust algorithms. In [16] Håstad proved that for **E3-LIN**( $\mathbf{G}$ ) (linear equations over an Abelian group  $\mathbf{G}$  where each equation contains precisely 3 variables) it is NP-hard to find an assignment satisfying  $(1/|\mathbf{G}| + \varepsilon)$ -fraction of the constraints given an instance which is  $(1 - \varepsilon)$ -satisfiable. Note that the trivial random algorithm achieves  $1/|\mathbf{G}|$  in expectation.

As observed in [30] the above results characterize robust solvability of all Boolean CSPs, because, by Schaefer’s theorem [28], **E3-LIN**( $\mathbf{G}$ ), **Horn- $k$ -Sat** and **2-SAT** are essentially the only CSPs with tractable decision problem. What about larger domains? A natural property which distinguishes **Horn- $k$ -Sat**, **2-SAT**, and **Unique-Games**( $q$ ) from **E3-LIN**( $\mathbf{G}$ ) and “algebraically” NP-complete CSPs is bounded width [12]. Briefly, a CSP has bounded width if the decision problem can be solved by checking local consistency of the instance. These problems were characterized independently by the authors [2, 3] and Bulatov [5]. It was proved that, in some sense, the only obstacle to bounded width is **E3-LIN**( $\mathbf{G}$ ) – the same problem which is difficult for robust satisfiability. These facts motivated Guruswami and Zhou to conjecture [15] that the class of bounded width CSPs coincide with the class of CSPs admitting a robust satisfiability algorithm.

Most of the recent developments in the decision version of the CSP are based on the algebraic approach introduced by Jeavons, Cohen and Gyssens [17] and refined by Bulatov, Krokhin and Jeavons [8, 6]. This approach was adjusted to work with robust solvability of CSPs in a recent paper by Dalmau and Krokhin [11]. As a consequence they proved one direction of the Guruswami–Zhou conjecture — if a CSP is robustly solvable then it necessarily has bounded width. They also proved the opposite direction in the special case of width 1 CSPs, and classified the robust solvability with respect to the rate of growth of the error function  $f$  in the Boolean case. Another recent paper by Kun, O’Donnell, Tamaki, Yoshida and Zhou [23] gives an independent proof for width 1 CSPs.

This paper confirms the Guruswami and Zhou conjecture in full generality. For any bounded width CSP we give a polynomial-time randomized algorithm for finding an assignment satisfying  $(1 - O(\log \log(1/\varepsilon)/\log(1/\varepsilon)))$ -fraction of constraints in expectation provided there exists an  $(1 - \varepsilon)$ -satisfying assignment (the presented derandomization achieves a worse ratio). The proof uncovers an interesting connection between the outputs of SDP (and LP) relaxations and Prague strategies – a consistency notion crucial for the bounded width characterization in [2, 3].

## 2. Preliminaries.

**2.1. CSP and robust algorithms.** We start by defining instances of the CSP.

DEFINITION 2.1. An instance of the CSP is a triple  $\mathcal{I} = (V, D, \mathcal{C})$  with  $V$  a finite set of variables,  $D$  a finite domain, and  $\mathcal{C}$  a finite list of constraints, where each constraint is a pair  $C = (S, R)$  with  $S$  a tuple of variables of length  $k$ , called the scope of  $C$ , and  $R$  a  $k$ -ary relation on  $D$  (i.e. a subset of  $D^k$ ), called the constraint relation of  $C$ .

An instance  $\mathcal{I}$  is trivial if all the constraint relations are empty.

An assignment for  $\mathcal{I}$  is a mapping  $F : V \rightarrow D$ . We say that  $F$  satisfies a constraint  $C = (S, R)$  if  $F(S) \in R$  (where  $F$  is applied component-wise). The value of  $F$ ,  $\text{Val}(F, \mathcal{I})$ , is the fraction of constraints it satisfies. The maximal value of  $\mathcal{I}$  is  $\text{Opt}(\mathcal{I}) = \max_{F: V \rightarrow D} \text{Val}(F, \mathcal{I})$ .

We study the CSP restricted to instances that use only relations from a fixed, finite set.

DEFINITION 2.2. A finite set of relations  $\Gamma$  on a finite set  $D$  is called a constraint language on  $D$ , and  $D$  is called the domain of  $\Gamma$ . An instance of  $\text{CSP}(\Gamma)$  is an instance of the CSP such that all the constraint relations are from  $\Gamma$ .

The decision problem for  $\text{CSP}(\Gamma)$  asks whether an input instance  $\mathcal{I}$  of  $\text{CSP}(\Gamma)$  has a solution, i.e. an assignment which satisfies all the constraints. The *Max-CSP* for  $\text{CSP}(\Gamma)$  asks to find an assignment of maximal value, i.e. such that  $\text{Val}(F, \mathcal{I}) = \text{Opt}(\mathcal{I})$ . This problem is computationally intractable for the vast majority of constraint languages motivating the study of approximation algorithms.

DEFINITION 2.3. Let  $\Gamma$  be a constraint language and let  $\alpha, \beta$  be real numbers. An algorithm  $(\alpha, \beta)$ -approximates  $\text{CSP}(\Gamma)$ , if it outputs an assignment  $F$  with  $\text{Val}(F, \mathcal{I}) \geq \alpha$  for every instance  $\mathcal{I}$  of  $\text{CSP}(\Gamma)$  such that  $\text{Opt}(\mathcal{I}) \geq \beta$ .

Our interest is in CSPs which can be well approximated on instances close to satisfiable.

DEFINITION 2.4. We say that  $\text{CSP}(\Gamma)$  is robustly solvable if there exists an error function  $g : [0, 1] \rightarrow [0, 1]$  such that  $\lim_{\varepsilon \rightarrow 0} g(\varepsilon) = 0$ , and a polynomial-time algorithm which  $(1 - g(\varepsilon), 1 - \varepsilon)$ -approximates  $\text{CSP}(\Gamma)$  for every  $\varepsilon \in [0, 1]$ .

**2.2. Bounded width.** Linear equations over an Abelian group are examples of CSPs which do not have bounded width. While the decision problem for these CSPs are tractable, they are not solvable by local propagation algorithms (unlike for example *Horn- $k$ -Sat*, *2-SAT*, or *Unique-Games( $q$ )*). A nice way to formalize solvability by local propagation is using the concept of a  $(k, l)$ -minimal instance. To do so we require a notion of projection: the projection of a constraint  $C$  to a tuple of variables  $x_1, \dots, x_m$  is a constraint on  $(x_1, \dots, x_m)$  with the constraint relation consisting of all  $(d_1, \dots, d_m)$ 's which can be extended to a tuple from the constraint relation of  $C$ .

DEFINITION 2.5. Let  $k \leq l$  be positive integers. An instance  $\mathcal{I} = (V, D, \mathcal{C})$  of the CSP is  $(k, l)$ -minimal, if:

- Every at most  $l$ -element tuple of distinct variables is within the scope of some constraint in  $\mathcal{C}$ ,
- For every tuple  $S$  of at most  $k$  distinct<sup>1</sup> variables and every pair of constraints  $C_1$  and  $C_2$  from  $\mathcal{C}$  whose scopes contain all variables from  $S$ , the projections

---

<sup>1</sup>Some technical problems in definitions can be caused by a variable appearing in a constraint more than once – they do not add to the complexity of the problem considered so we disregard them here.

to  $S$  are the same. This projection is denoted by  $P_S^{\mathcal{I}}$ , or  $P_S$ .

A  $(k, k)$ -minimal instance is also called  $k$ -minimal.

For fixed  $k, l$  there is an obvious polynomial-time algorithm for transforming an instance  $\mathcal{I}$  of the CSP to a  $(k, l)$ -minimal instance with the same set of solutions: First we add new constraints (initially allowing all the evaluations) to ensure that the first condition is satisfied and then we gradually remove those tuples from the constraint relations which falsify the second condition. We call the resulting instance *the  $(k, l)$ -minimal instance corresponding to  $\mathcal{I}$* . The definite article is justified since it is easy to see that the obtained instance is independent on the precise order of removals. It is clear that if the  $(k, l)$ -minimal instance corresponding to  $\mathcal{I}$  is trivial then the original instance had no solution. We say that  $\text{CSP}(\Gamma)$  has width  $(k, l)$  if the converse is always true.

**DEFINITION 2.6.** *Let  $k \leq l$  be positive integers and let  $\Gamma$  be a constraint language. We say that  $\text{CSP}(\Gamma)$  has width  $(k, l)$  if every instance  $\mathcal{I}$  of  $\text{CSP}(\Gamma)$ , whose corresponding  $(k, l)$ -minimal instance is nontrivial, has a solution.*

*We say that  $\text{CSP}(\Gamma)$  (or  $\Gamma$ ) has bounded width if it has width  $(k, l)$  for some  $k, l$ .*

Different notions of width are often used in the literature, but they all lead to equivalent concepts of bounded width. We refer to [12, 26, 7] for formal definitions and background.

**2.3. Primitive positive definitions, polymorphisms.** Primitive positive definitions are very useful in the decision version of CSP. We say that a relation  $R$  on  $D$  is *primitively positively definable* (or just *pp-definable*) from a constraint language  $\Gamma$  if there exists a (primitive positive) formula

$$\phi(x_1, \dots, x_k) \equiv \exists y_1, \dots, y_l \psi(x_1, \dots, x_k, y_1, \dots, y_l) ,$$

where  $\psi$  is a conjunction of atomic formulas using relations in  $\Gamma$  and the (binary) equality relation on  $D$  such that

$$(a_1, \dots, a_k) \in R \text{ if and only if } \phi(a_1, \dots, a_k) \text{ holds .}$$

The algebraic approach to the CSP is based on a theorem by Geiger [13] and also by Bodarchuk et al. [4] which says that pp-definability is in the sense of Theorem 2.8 controlled by certain operations called polymorphisms. We will discuss the impact of particular polymorphisms on complexity of CSP in next sections.

**DEFINITION 2.7.** *An  $l$ -ary operation  $f$  on  $D$  (i.e. a mapping  $f : D^l \rightarrow D$ ) is compatible with a  $k$ -ary relation  $R$ , if*

$$(f(a_1^1, \dots, a_1^l), f(a_2^1, \dots, a_2^l), \dots, f(a_k^1, \dots, a_k^l)) \in R$$

*whenever  $(a_1^1, \dots, a_k^1), (a_1^2, \dots, a_k^2), \dots, (a_1^l, \dots, a_k^l) \in R$ .*

*We say that  $f$  is a polymorphism of a constraint language  $\Gamma$ , if it is compatible with every relation in  $\Gamma$ . The set of all polymorphisms of  $\Gamma$  will be denoted by  $\text{Pol}(\Gamma)$ .*

An  $n$ -ary relation  $R$  is *irredundant* if for every pair of different coordinates  $1 \leq i < j \leq n$ , the relation  $R$  contains a tuple  $(a_1, a_2, \dots, a_n) \in R$  with  $a_i \neq a_j$ . The following theorem ties the notion of pp-definitions and polymorphisms together.

**THEOREM 2.8.** [13, 4] *Let  $\Gamma$  be a constraint language on  $D$  and let  $R$  be a nonempty relation on  $D$ . Then  $\text{Pol}(\Gamma) \subseteq \text{Pol}(R)$  if and only if  $R$  is pp-definable from  $\Gamma$ . Moreover, if  $R$  is irredundant and  $\text{Pol}(\Gamma) \subseteq \text{Pol}(R)$  then  $R$  is pp-definable from  $\Gamma$  without equality.*

**3. The conjecture and known reductions.** The conjecture of Guruswami and Zhou [15] states

CONJECTURE OF GURUSWAMI AND ZHOU. *Let  $\Gamma$  be a constraint language. The following are equivalent:*

- $\text{CSP}(\Gamma)$  has bounded width;
- $\text{CSP}(\Gamma)$  is robustly solvable.

The upward implication in the conjecture was proved by Dalmau and Krokhin [11] (assuming  $P \neq NP$ ) by combining the characterization of problems of bounded width [2, 3, 5] with a result of Håstad [16]. Their proof uses an adjustment to the algebraic approach (developed by its authors) which is usually used for the decision version of CSP. This paper proves the downward direction of the conjecture.

**3.1. Primitive positive definitions.** An important observation for decision CSPs is that we do not increase the complexity (modulo log-space reductions) by adding a pp-definable relation to the constraint language [17]. More importantly, from the point of view of this article, adding pp-definable relations into the constraint language does not change the property of having bounded width [25, 26].

A similar fact was proved in for robust solvability [11], under additional assumption that the pp-definition does not involve the equality relation. To state the result concisely we introduce the notation  $\text{CSP}(\Gamma) \leq_{RA} \text{CSP}(\Gamma')$  as a shorthand for: for any error function  $f$  with  $\lim_{\varepsilon \rightarrow 0} f(\varepsilon) = 0$ , if some polynomial-time algorithm  $(1 - f(\varepsilon), 1 - \varepsilon)$ -approximates  $\text{CSP}(\Gamma')$  for every  $\varepsilon \geq 0$  then there exists a polynomial-time algorithm that  $(1 - O(f(\varepsilon)), 1 - \varepsilon)$ -approximates  $\text{CSP}(\Gamma)$  for every  $\varepsilon \geq 0$ .

THEOREM 3.1 ([11]). *Let  $\Gamma$  be a constraint language on  $D$  and let  $R$  be a relation on  $D$ . If  $R$  is pp-definable from  $\Gamma$  without equality, then  $\text{CSP}(\Gamma \cup \{R\}) \leq_{RA} \text{CSP}(\Gamma)$ .*

As the relations pp-definable in  $\Gamma$  are fully determined by polymorphisms of  $\Gamma$ , the complexity of the decision problem as well as the property of having bounded width for  $\text{CSP}(\Gamma)$  depends only on the algebraic structure of  $\text{Pol}(\Gamma)$ . Robust solvability (including the order of the error function) is also “to a large extent” controlled by  $\text{Pol}(\Gamma)$ . Unfortunately, we have to say “to a large extent” because of the disturbing fact that Theorem 3.1 allows only pp-definitions without equality. The general case with equality is open.

**3.2. Cores and singleton expansions.** Another important observation for both decision CSPs and robust solvability of CSPs is that we can restrict our attention to cores.

DEFINITION 3.2. *We say that a constraint language is a core, if all its unary polymorphisms are bijections.*

If  $\Gamma$  is a constraint language on  $D$  which is not a core we can define another constraint language  $\Gamma'$  on a smaller domain such that  $\text{CSP}(\Gamma)$  and  $\text{CSP}(\Gamma')$  behave identically with respect to decision, approximation and have the same width. Namely, if  $e$  is a non-surjective unary polymorphism of  $\Gamma$  then we define  $\Gamma' = \{e(R) : R \in \Gamma\}$ , where  $e(R) = \{(e(a_1), \dots, e(a_n)) : (a_1, \dots, a_n) \in R\}$  (see [11] for more details).

A nontrivial fact is that we can add singleton unary relations to any core language  $\Gamma$  without significantly changing robust solvability, complexity of the decision problem or property of having bounded width for  $\text{CSP}(\Gamma)$ .

THEOREM 3.3 ([6, 11]). *Let  $\Gamma$  be a core constraint language and let  $\Gamma' = \Gamma \cup \{\{a\} : a \in D\}$ , then:*

- $\text{CSP}(\Gamma)$  and  $\text{CSP}(\Gamma')$  are log-space equivalent,
- $\text{CSP}(\Gamma') \leq_{RA} \text{CSP}(\Gamma) \leq_{RA} \text{CSP}(\Gamma')$ , and
- $\text{CSP}(\Gamma)$  has bounded width if and only if  $\text{CSP}(\Gamma')$  does.

We refer to  $\Gamma'$  from this theorem as the *singleton expansion* of  $\Gamma$ . Theorem 3.3 implies that the characterization conjectured by Guruswami and Zhou needs to be verified for singleton expansions of constraint languages only. This restricts the family of constraint languages one needs to consider and we use this fact repeatedly throughout the paper.

Another consequence of Theorem 3.3 is that whenever  $\text{CSP}(\Gamma)$  is tractable then there is a polynomial-time algorithm for finding a solution.

**THEOREM 3.4.** [6] *Let  $\Gamma$  be a constraint language such that the decision problem for  $\text{CSP}(\Gamma)$  is solvable in a polynomial time. Then there is a polynomial-time algorithm for finding a solution of  $\text{CSP}(\Gamma)$*

A proof of this theorem uses the singleton unary relations in the singleton expansion of the core of  $\Gamma$  to recursively set values for variables and verify if such a partial evaluation extends to a solution.

**3.3. Interpretations.** Primitive positive definitions can be used to compare the constraint languages on the same domain. A stronger tool which also enables us to compare CSPs on different domains are pp-interpretations.

**DEFINITION 3.5.** *Let  $\Gamma$  be a constraint languages and:*

- $U$  be a pp-definable relation in  $\Gamma$ ,
- $\Theta$  be an equivalence relation on  $U$  pp-definable<sup>2</sup> in  $\Gamma$ ,
- $S_i$  be relations on  $U$  pp-definable<sup>3</sup> in  $\Gamma$ .

*The language  $\Gamma' = \{S_i/\Theta = \{(u_1/\Theta, \dots, u_{n_i}/\Theta) : (u_1, \dots, u_{n_i}) \in S_i\}\}_i$  on the domain  $U/\Theta$  (and every language isomorphic to it) is pp-interpretable in  $\Gamma$ .*

It was proved in [6, 24] (using a slightly different language) that if  $\Gamma'$  is pp-interpretable in  $\Gamma$  then the decision problem for  $\text{CSP}(\Gamma')$  is log-space reducible to the decision problem for  $\text{CSP}(\Gamma)$ ; moreover if  $\text{CSP}(\Gamma)$  has bounded width then so does  $\text{CSP}(\Gamma')$  [25, 26].

A similar theorem, in a more restrictive setting, was proved for robust solvability [11]. In this setting the relation  $U$  needs to be unary, and the pp-definitions of  $\Theta$  and  $S_i$ 's cannot use equality.

**3.4. The hardness result.** A proof of the hardness part of the characterization [11] is based on a theorem by Håstad [16], in which he establishes hardness for particular CSPs connected to Abelian groups.

For a finite Abelian group  $\mathbf{G} = (G, +)$  let  $\Gamma(\mathbf{G})$  denotes the constraint language on the domain  $D = G$  consisting of all relations encoding linear equations over  $\mathbf{G}$  with 3 variables, that is, relations of the form  $\{(x, y, z) \in G^3 : ax + by + cz = d\}$  for some  $d \in G, a, b, c \in \mathbb{Z}$ . The corresponding  $\text{CSP}(\Gamma(\mathbf{G}))$  is denoted by **E3-LIN**( $\mathbf{G}$ ).

**THEOREM 3.6** ([16]). *If  $\mathbf{G}$  is an Abelian group with  $n > 1$  elements then for every  $\varepsilon > 0$  there is no polynomial-time algorithm that  $(1/n + \varepsilon, 1 - \varepsilon)$ -approximates  $\text{CSP}(\Gamma(\mathbf{G}))$  unless  $\text{P} = \text{NP}$ .*

Turning to equivalent descriptions of problems of bounded width we restrict our attention (using the results of subsection 3.2) to singleton expansions of languages. Combining the results of [12, 26, 2, 3, 1, 5] we obtain

**THEOREM 3.7.** *Let  $\Gamma$  be a singleton expansion of a constraint language. The following are equivalent.*

- (a) *There does not exist a nontrivial Abelian group  $\mathbf{G}$  such that  $\Gamma(\mathbf{G})$  is pp-interpretable in  $\Gamma$ .*

<sup>2</sup>Throughout the definition we identify  $U^2$  with an appropriate power of the domain of  $\Gamma$ .

<sup>3</sup>Similarly we identify powers of  $U$  with (usually higher) powers of the domain of  $\Gamma$ .



- (b)  $\text{CSP}(\Gamma)$  has bounded width.
- (c)  $\text{CSP}(\Gamma)$  has width  $(2, 3)$ .
- (d)  $\text{Pol}(\Gamma)$  contains a 3-ary operation  $f_1$  and a 4-ary operation  $f_2$  such that, for all  $a, b \in D$ ,

$$\begin{aligned} f_1(a, a, b) &= f_1(a, b, a) = f_1(b, a, a) = \\ &= f_2(a, a, a, b) = \dots = f_2(b, a, a, a) \end{aligned}$$

and  $f_1(a, a, a) = a$ .

A refinement of condition (a) in the previous theorem is needed for robust solvability (see [11] for more detailed discussion and references):

**THEOREM 3.8.** *Let  $\Gamma$  be a singleton expansion of a constraint language. The following condition is equivalent to conditions from Theorem 3.7*

- (e) *There does not exist a nontrivial Abelian group  $\mathbf{G}$  such that  $\Gamma(\mathbf{G})$  is pp-interpretable in  $\Gamma$  in the first power of the domain and using pp-definitions without equality.*

Combining this fact with the results of [11] discussed in subsection 3.3 and Theorem 3.6 the authors of [11] obtain the hardness proof, i.e. the upward direction of the conjecture of Guruswami and Zhou (unless  $\text{P}=\text{NP}$ ).

**3.5. The missing implication.** The main result of this paper proves the missing implication and therefore confirms the conjecture of Guruswami and Zhou. As discussed in subsection 3.2 we can, without loss of generality, assume that  $\Gamma$  is a singleton expansion of a constraint language. This is a statement of the main theorem in the paper:

**THEOREM 3.9.** *If  $\Gamma$  is a singleton expansion of a constraint language and the  $\text{CSP}(\Gamma)$  has bounded width then it is robustly solvable. More precisely there exists a randomized polynomial-time algorithm which returns an assignment satisfying, in expectation,  $(1 - O(\log \log(1/\varepsilon)/\log(1/\varepsilon)))$ -fraction of the constraints given a  $(1 - \varepsilon)$ -satisfiable instance.*

**3.6. An overview of the proof.** Efficient approximation algorithms are often designed through linear programming (LP) relaxations and semidefinite programming (SDP) relaxations. For instance, the robust satisfiability algorithm for **Horn- $k$ -Sat** [30] uses LP relaxation while the robust satisfiability algorithms for **2-SAT** and **Unique-Games( $q$ )** [30, 10] are SDP-based.

Robust algorithms for all CSPs of width 1 were independently devised in [11] and [23]. From the CSPs mentioned previously, this result covers **Horn- $k$ -Sat**, but not **2-SAT** or **Unique-Games( $q$ )**. The approach in [23] is close to ours so let us briefly sketch the main ideas.

For any instance  $\mathcal{I} = (V, D, \mathcal{C})$  there is a canonical 0–1 integer program with the same optimal value as Max-CSP. It has variables  $\lambda_x(a)$  for every  $x \in V$  and  $a \in D$  and variables  $\lambda_C(\mathbf{a})$  for every constraint  $C = (S, R)$  and every tuple  $\mathbf{a} \in A^r$ , where  $r$  is the arity of  $C$ . The interpretation of  $\lambda_x(a) = 1$  is that variable  $x$  is assigned value  $a$ ; the interpretation of  $\lambda_C(\mathbf{a}) = 1$  is that  $S$  is assigned (component-wise) tuple  $\mathbf{a}$ . The value to be maximized is then equal to

$$\frac{1}{|\mathcal{C}|} \sum_{C=(S,R) \in \mathcal{C}} \sum_{\mathbf{a} \in R} \lambda_C(\mathbf{a}). \quad (3.1)$$

modulo the following constraints

$$\begin{aligned} \sum_{a \in D} \lambda_x(a) &= 1 \text{ for every } x \in V \\ \sum_{\mathbf{a}: a_i = a} \lambda_C(\mathbf{a}) &= \lambda_{x_i}(a) \text{ for every } C = ((x_1, \dots, x_r), R), i \leq r \text{ and } a \in D. \end{aligned}$$

By relaxing the 0–1 program allowing the variables to take values in the range  $[0, 1]$  instead of  $\{0, 1\}$ , we obtain the *basic linear programming relaxation* for  $\mathcal{I}$  with possibly larger value  $\text{LPOpt}(\mathcal{I})$  of the sum (3.1).

The robust algorithm from [23] works roughly as follows. (1) Run the basic LP relaxation for  $\mathcal{I}$ , (2) use the output of LP to remove some constraints so that the remaining instance  $\mathcal{J}$  has the property that the 1-minimal instance corresponding to  $\mathcal{J}$  is non-trivial, (3) return a solution of  $\mathcal{J}$ . Steps (1) and (2) can be performed on any instance of the CSP. The instance  $\mathcal{J}$  after step (2) has a solution whenever the language has width 1, therefore we can perform step (3) using, for instance, Theorem 3.4.

Our robust algorithm for all bounded width CSPs has the same general form. The differences are that we use it only for instances with at most binary constraints (a reduction is provided in the next section). In step (1) we use the basic SDP relaxation instead of the basic LP relaxation, and in step (2) we use weak Prague instances (see Section 5).

#### 4. Our tools and reductions.

**4.1. Reduction to constraint languages with unary and binary relations.** In this section we present a reduction which allows us to prove Theorem 3.9 in an even more restricted setting: for singleton expansions of constraint languages with unary and binary constraints only. The reduction is given in the following proposition.

**PROPOSITION 4.1.** *Let  $\Gamma$  be a singleton expansion of a constraint language on the domain  $D$  which contains relations of maximum arity  $l$  and such that  $\text{CSP}(\Gamma)$  has bounded width. Then there exists  $\Gamma'$  a singleton expansion of a constraint language on  $D'$  containing only at most binary relations such that  $\text{CSP}(\Gamma')$  has bounded width and  $\text{CSP}(\Gamma) \leq_{\text{RA}} \text{CSP}(\Gamma')$ .*

*Proof.* First we define the constraint language  $\Gamma'$  on  $D' = D^l$ . For every relation  $R \in \Gamma$  of arity  $k$  we add to  $\Gamma'$  the unary relation  $R'$  defined by

$$(a_1, \dots, a_l) \in R' \quad \text{iff} \quad (a_1, \dots, a_k) \in R,$$

for every  $k \leq l$  we add the binary relation

$$E_k = \{((a_1, \dots, a_l), (b_1, \dots, b_l)) : a_1 = b_k\},$$

and for every  $(a_1, \dots, a_l) \in D'$  we add the singleton unary relation  $\{(a_1, \dots, a_l)\}$ . The singletons ensure that  $\Gamma'$  is a singleton expansion. The  $\text{CSP}(\Gamma')$  has bounded width which can be seen, for instance, from Theorem 3.7: If  $f_1, f_2$  are polymorphisms of  $\Gamma$  from this theorem, then the corresponding operations  $f'_1, f'_2$  acting coordinate-wise on  $D'$  satisfy the same equations and it is straightforward to check that  $f'_1, f'_2$  are polymorphisms of  $\Gamma'$ .

Now, let  $\mathcal{I} = (V, D, \mathcal{C})$  be an instance of  $\text{CSP}(\Gamma)$  with  $\text{Opt}(\mathcal{I}) = 1 - \varepsilon$ . We transform  $\mathcal{I}$  to an instance  $\mathcal{I}'$  of  $\text{CSP}(\Gamma')$  as follows. We keep the original variables



and for every constraint  $C = ((x_1, \dots, x_k), R)$  in  $\mathcal{C}$  we introduce a new variable  $x_C$  and add  $k + 1$  constraints

$$((x_C), R'), ((x_1, x_C), E_1), ((x_2, x_C), E_2), \dots, ((x_k, x_C), E_k). \quad (4.1)$$

If  $F : V \rightarrow D$  is an assignment for  $\mathcal{I}$  of value  $1 - \varepsilon$  then the assignment  $F'$  for  $\mathcal{I}'$  defined by

$$\begin{aligned} F'(x) &= (F(x), ?, \dots, ?) \quad \text{for } x \in V, \\ F'(x_C) &= (F(x_1), \dots, F(x_k), ?, \dots, ?) \\ &\quad \text{for } C = ((x_1, \dots, x_k), R) \end{aligned}$$

(where  $?$  stands for an arbitrary element of  $D$ ) has value at least  $1 - \varepsilon$  since all the binary constraints in  $\mathcal{I}'$  are satisfied and the constraint  $(x_C, R')$  is satisfied whenever  $F$  satisfies  $C$ .

We run the robust algorithm for  $\text{CSP}(\Gamma')$  to get an assignment  $G'$  for  $\mathcal{I}'$  with value at least  $1 - g(\varepsilon)$ , and we define  $G(x)$ ,  $x \in V$  to be the first coordinate of  $G'(x)$ . Note that, for any constraint  $C$  of  $\mathcal{I}$ , if  $G'$  satisfies all the constraints (4.1) then  $G$  satisfies  $C$ . Therefore the value of  $G$  is at least  $1 - (l + 1)g(\varepsilon)$ .  $\square$

Now to prove Theorem 3.9 for  $\Gamma$  – a singleton expansion of an arbitrary constraint language, we produce  $\Gamma'$  (from Lemma 4.1) and if Theorem 3.9 holds for  $\Gamma'$  (which has at most binary constraints) it does for  $\Gamma$  as well.

**4.2. LP and SDP relaxations.** The previous subsection allows us to present a simplified version of the definition of a basic SDP relaxation [27] which is appropriate for languages with only unary and binary constraints.

**DEFINITION 4.2.** *Let  $\Gamma$  be a constraint language over  $D$  consisting of at most binary relations and let  $\mathcal{I} = (V, D, \mathcal{C})$  be an instance of  $\text{CSP}(\Gamma)$  with  $m$  constraints. The goal for the basic SDP relaxation of  $\mathcal{I}$  is to find  $(|V||D|)$ -dimensional real vectors  $\mathbf{x}_a$ ,  $x \in V$ ,  $a \in D$  maximizing*

$$\frac{1}{m} \left( \sum_{(x,R) \in \mathcal{C}} \sum_{a \in R} \|\mathbf{x}_a\|^2 + \sum_{((x,y),R) \in \mathcal{C}} \sum_{(a,b) \in R} \mathbf{x}_a \mathbf{y}_b \right) \quad (4.2)$$

subject to

- (SDP1)**  $\mathbf{x}_a \mathbf{y}_b \geq 0$  for all  $x, y \in V$ ,  $a, b \in D$
- (SDP2)**  $\mathbf{x}_a \mathbf{x}_b = 0$  for all  $x \in V$ ,  $a, b \in D$ ,  $a \neq b$ , and
- (SDP3)**  $\sum_{a \in D} \mathbf{x}_a = \sum_{a \in D} \mathbf{y}_a$ ,  $\|\sum_{a \in D} \mathbf{x}_a\|^2 = 1$   
for all  $x, y \in V$ .

The dot products  $\mathbf{x}_a \mathbf{y}_b$  can be thought of as weights and the goal is to find vectors so that maximum weight is given to pairs (or elements) in constraint relations. It will be convenient to use the notation

$$\mathbf{x}_A = \sum_{a \in A} \mathbf{x}_a$$

for a variable  $x \in V$  and a subset  $A \subseteq D$ , so that condition (SDP3) can be written as  $\mathbf{x}_D = \mathbf{y}_D$ ,  $\|\mathbf{x}_D\|^2 = 1$ . The contribution of one constraint to (4.2) is by (SDP3) at most 1 and it is the greater the less weight is given to pairs (or elements) outside the constraint relation.

The optimal value for the sum (4.2),  $\text{SDPOpt}(\mathcal{I})$ , is always at least  $\text{Opt}(\mathcal{I})$ . There are algorithms (see e.g. [29]) that output vectors with  $(4.2) \geq \text{SDPOpt}(\mathcal{I}) - \delta$  which are polynomial in the input size and  $\log(1/\delta)$ .

From the output of the basic SDP relaxation we can get a valid output of the LP relaxation by defining  $\lambda_x(a) = \|\mathbf{x}_a\|^2$  and  $\lambda_{(x,y)}(a,b) = \mathbf{x}_a \mathbf{y}_b$  for any constraint  $((x,y), R)$ . In particular,  $\text{SDPOpt}(\mathcal{I}) \leq \text{LPOpt}(\mathcal{I})$ .

**5. Prague instances.** The proof of the characterization of bounded width CSPs in [2] relies on a certain consistency notion called Prague strategy. It turned out that Prague strategies are related to outputs of basic SDP relaxations and this connection is what made our main result possible. The main result actually uses a stronger result, about weaker consistency notion called weak Prague instance [3].

Terms defined below are used only for certain types of instances and constraint languages. In our main proof we will construct, using an output of an SDP program, a weak Prague instance in a language which is different than the language of the original instance. Therefore, in the remainder of this section we assume that

- $\Lambda$  is a constraint language on a domain  $D$ ,  $\Lambda$  contains only binary relations,
- $\mathcal{J} = (V, D, \mathcal{C}^{\mathcal{J}})$  is an instance of  $\text{CSP}(\Lambda)$  such that every pair of distinct variables is the scope of at most one constraint  $((x,y), P_{x,y}^{\mathcal{J}})$ , and if  $((x,y), P_{x,y}^{\mathcal{J}}) \in \mathcal{C}^{\mathcal{J}}$  then  $((y,x), P_{y,x}^{\mathcal{J}}) \in \mathcal{C}^{\mathcal{J}}$ , where  $P_{y,x}^{\mathcal{J}} = \{(b,a) : (a,b) \in P_{x,y}^{\mathcal{J}}\}$ . (Usually the instance is clear from context and then we omit the superscripts for  $P_{x,y}$ 's and  $\mathcal{C}$ .)

Note that under these assumptions  $\mathcal{J}$  is 1-minimal if and only if every variable is in the scope of some constraint and for every constraint  $((x,y), P_{x,y}^{\mathcal{J}})$  the projection of  $P_{x,y}^{\mathcal{J}}$  to the first coordinate is equal to  $P_x^{\mathcal{J}}$ , where  $P_x^{\mathcal{J}}$  are the sets from the definition of 1-minimality.

**5.1. Weak Prague instance.** First we need to define steps and patterns.

**DEFINITION 5.1.** A step (in  $\mathcal{J}$ ) is a pair of variables  $(x,y)$  which is the scope of a constraint in  $\mathcal{J}$ . A pattern from  $x$  to  $y$  is a sequence of variables  $p = (x = x_1, x_2, \dots, x_k = y)$  such that every  $(x_i, x_{i+1})$ ,  $i = 1, \dots, k-1$  is a step.

For a pattern  $p = (x_1, \dots, x_k)$  we put  $-p = (x_k, \dots, x_1)$ . If  $p = (x_1, \dots, x_k)$ ,  $q = (y_1, \dots, y_l)$ ,  $x_k = y_1$  then the concatenation of  $p$  and  $q$  is the pattern  $p + q = (x_1, x_2, \dots, x_k = y_1, y_2, \dots, y_l)$ . For a pattern  $p$  from  $x$  to  $x$  and a natural number  $k$ ,  $kp$  denotes the  $k$ -time concatenation of  $p$  with itself.

Observe that from the assumptions about  $\mathcal{J}$  it follows that  $-p$  is a pattern whenever  $p$  is.

**DEFINITION 5.2.** Let  $p = (x = x_1, x_2, \dots, x_k = y)$  be a pattern from  $x$  to  $y$  in  $\mathcal{J}$ . A realization of  $p$  is a sequence  $(a_1, \dots, a_k) \in D^k$  such that  $(a_i, a_{i+1}) \in P_{x_i, x_{i+1}}$  for every  $1 \leq i \leq k-1$ .

For a subset  $A \subseteq D$  we define  $A + p$  as the set of the last elements of those realizations of  $p$  whose first element is in  $A$ , that is,

$$A + p = \{b \in D : (\exists a_1, \dots, a_{k-1} \in D) a_1 \in A \text{ and } (a_1, \dots, a_{k-1}, b) \text{ is a realization of } p\}.$$

Finally, we define  $A - p = A + (-p)$ .

The addition of patterns is associative, i.e.  $(A + p) + q = A + (p + q)$ . Also note that in a 1-minimal instance we have  $A \subseteq A + p - p$  for any  $A \subseteq P_x$  and any pattern  $p$  from  $x$ .

A weak Prague instance is a 1-minimal instance with additional requirements concerning addition of patterns.

**DEFINITION 5.3.**  $\mathcal{J}$  is a weak Prague instance if

- (P1)  $\mathcal{J}$  is 1-minimal,  
(P2) for every  $A \subseteq P_x^{\mathcal{J}}$  and every pattern  $p$  from  $x$  to  $x$ , if  $A + p = A$  then  $A - p = A$ , and  
(P3) for any patterns  $p_1, p_2$  from  $x$  to  $x$  and every  $A \subseteq P_x^{\mathcal{J}}$ , if  $A + p_1 + p_2 = A$  then  $A + p_1 = A$ .

To clarify the definition let us assume that  $\mathcal{J}$  is 1-minimal and consider the following digraph: vertices are all the pairs  $(A, x)$ , where  $x \in V$  and  $A \subseteq P_x^{\mathcal{J}}$ , and  $((A, x), (B, y))$  forms an edge iff  $(x, y)$  is a step and  $A + (x, y) = B$ . Condition (P3) means that no strong component contains  $(A, x)$  and  $(A', x)$  with  $A \neq A'$ , condition (P2) is equivalent (by the following lemma) to the fact that every strong component contains only undirected edges (that is, if  $((A, x), (B, y))$  is an edge then so is  $((B, y), (A, x))$ ).

LEMMA 5.4. *Let  $\mathcal{J}$  be a 1-minimal instance. Then (P2) is equivalent to the following condition.*

- (P2\*) For every step  $(x, y)$ , every  $A \subseteq P_x$  and every pattern  $p$  from  $y$  to  $x$ , if  $A + (x, y) + p = A$  then  $A + (x, y, x) = A$ .

*Proof.* (P2\*)  $\Rightarrow$  (P2). If  $p = (x = x_1, x_2, \dots, x_k = x)$  is a pattern from  $x$  to  $x$  such that  $A + p = A$ , then repeated application of (P2\*) gives us

$$\begin{aligned}
A + p - p &= \\
&= [A + (x_1, x_2, \dots, x_{k-1})] + (x_{k-1}, x_k, x_{k-1}) \\
&\quad + (x_{k-1}, x_{k-2}, \dots, x_1) = \\
&= A + (x_1, x_2, \dots, x_{k-1}) + (x_{k-1}, x_{k-2}, \dots, x_1) = \\
&= [A + (x_1, x_2, \dots, x_{k-2})] + (x_{k-2}, x_{k-1}, x_{k-2}) \\
&\quad + (x_{k-2}, x_{k-3}, \dots, x_1) = \\
&= A + (x_1, x_2, \dots, x_{k-2}) + (x_{k-2}, x_{k-3}, \dots, x_1) = \\
&= \dots = \\
&= A,
\end{aligned}$$

where the second equality uses (P2\*) for the set  $A + (x_1, x_2, \dots, x_{k-1})$ . The assumption of (P2\*) is provided by a cyclic shift of the pattern  $p$ :  $[A + (x_1, \dots, x_{k-1})] + (x_{k-1}, x_k) + (x_1, \dots, x_k, \dots, x_{k-1}) = [A + (x_1, x_2, \dots, x_{k-1})]$  as  $A + (x_1, \dots, x_{k-1}) + (x_{k-1}, x_k) = A$ . The fourth equality uses (P2\*) for the set  $A + (x_1, \dots, x_{k-2})$  and so on.

(P2)  $\Rightarrow$  (P2\*). By applying (P2) to the pattern  $(x, y) + p$  we get  $A + (x, y) + p - p + (y, x) = A$ . From 1-minimality it follows that  $A + (x, y) \subseteq A + (x, y) + p - p$ , hence  $A + (x, y, x) = (A + (x, y)) + (y, x) \subseteq (A + (x, y) + p - p) + (y, x) = A$ . The other inclusion follows again from 1-minimality.  $\square$

EXAMPLE 1. *An example of a weak Prague instance, which is not a Prague strategy [3] i.e. witnessing that the new notion is weaker, is  $V = \{x, y, z\}$ ,  $D = \{0, 1\}$ ,  $P_{x,y} = P_{x,z} = \{(0, 0), (1, 1)\}$ ,  $P_{y,z} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ .*

*If we change  $P_{y,z}$  to  $\{(0, 1), (1, 0)\}$  the conditions (P1) and (P2) hold but  $\{0\} + (x, y, z, x) + (x, y, z, x) = \{0\}$  and  $\{0\} + (x, y, z, x) = \{1\}$ .*

*If, on the other hand, we set  $P_{y,z}$  to  $\{(0, 0), (1, 0), (1, 1)\}$  then (P1) and (P3) hold while  $\{0\} + (x, y, z, x) = \{0\}$ , but  $\{0\} - (x, y, z, x) = \{0, 1\}$ .*

The main result of this paper relies on the following theorem.

THEOREM 5.5 ([3]). *If  $\text{CSP}(\Lambda)$  has bounded width and  $\mathcal{J}$  is a nontrivial weak Prague instance of  $\text{CSP}(\Lambda)$  then  $\mathcal{J}$  has a solution.*

**5.2. SDP and Prague instances.** We now show that one can naturally associate a weak Prague instance to an output of the basic SDP relaxation. This material will not be used in what follows, it is included to provide some intuition for the proof of the main theorem.

Let  $\mathbf{x}_a, x \in V, a \in D$  be arbitrary vectors satisfying (SDP1), (SDP2) and (SDP3). We define a CSP instance  $\mathcal{J}$  by

$$\mathcal{J} = (V, D, \{((x, y), P_{x,y}) : x, y \in V, x \neq y\}), \\ P_{x,y} = \{(a, b) : \mathbf{x}_a \mathbf{y}_b > 0\},$$

and we show that it is a weak Prague instance.

The instance is 1-minimal with  $P_x^{\mathcal{J}} = \{a \in D : \mathbf{x}_a \neq \mathbf{0}\}$ . To prove this it is enough to verify that the projection of  $P_{x,y}$  to the first coordinate is equal to  $P_x^{\mathcal{J}}$ . If  $(a, b) \in P_{x,y}$ , then clearly  $\mathbf{x}_a$  cannot be the zero vector, therefore  $a \in P_x^{\mathcal{J}}$ . On the other hand, if  $a \in P_x^{\mathcal{J}}$  then  $0 < \|\mathbf{x}_a\|^2 = \mathbf{x}_a \mathbf{x}_D = \mathbf{x}_a \mathbf{y}_D$  and thus at least one of the dot products  $\mathbf{x}_a \mathbf{y}_b, b \in D$  is nonzero and  $(a, b) \in P_{x,y}$ .

To check (P2) and (P3) we note that, for any  $x, y \in V, x \neq y$  and  $A \subseteq P_x^{\mathcal{J}}$ , the vector  $\mathbf{y}_{A+(x,y)}$  has either a strictly greater length than  $\mathbf{x}_A$ , or  $\mathbf{x}_A = \mathbf{y}_{A+(x,y)}$ , and the latter happens iff  $A + (x, y, x) = A$  (see the proof of Claim 4, in fact, one can check that  $\mathbf{y}_{A+(x,y)}$  is obtained by adding to  $\mathbf{x}_A$  an orthogonal vector whose size is strictly greater than zero iff  $A + (x, y, x) \neq A$ ). By induction, for any pattern  $p$  from  $x$  to  $y$ , the vector  $\mathbf{y}_{A+p}$  is either strictly longer than  $\mathbf{x}_A$ , or  $\mathbf{x}_A = \mathbf{y}_{A+p}$  and  $A + p - p = A$ . Now (P2) follows immediately and (P3) is also easily seen: If  $A + p + q = A$  then necessarily  $\mathbf{x}_A = \mathbf{x}_{A+p}$  which is possible only if  $A = A + p$ .

We end this section with several remarks.

**5.2.1. Considering only the squares of length of vectors is equivalent to LP.** To prove property (P2) we only need to consider the lengths of the vectors. In fact, this property will be satisfied when we start with the basic linear programming relaxation (and define the instance  $\mathcal{J}$  in a similar way — compare the end of section 4.2). This is not the case for property (P3).

**5.2.2. This is a Prague strategy.** The above weak Prague instance is in fact a Prague strategy in the sense of [2]. This means that every pair of variables is the scope of a (unique) constraint and all strong components of the digraph introduced after Definition 5.3 are complete graphs.

**5.2.3. The SDP relaxation does not guarantee a (2, 3)-minimal instance.** There were attempts to show that the instance  $\mathcal{J}$  is (2, 3)-minimal after adding appropriate ternary constraints. This is equivalent to the requirement that  $P_{x,y}$  is a subset of the composition of the relations  $P_{x,z}$  and  $P_{z,y}$  for every  $x, y, z$ . The following example shows that it is not the case. Consider  $V = \{x, y, z\}$ ,  $D = \{0, 1\}$  and vectors  $\mathbf{x}_0 = (1/2, 1/2, 0)$ ,  $\mathbf{x}_1 = (1/2, -1/2, 0)$ ,  $\mathbf{y}_0 = (1/4, -1/4, \sqrt{2}/4)$ ,  $\mathbf{y}_1 = (3/4, 1/4, -\sqrt{2}/4)$ ,  $\mathbf{z}_0 = (1/4, 1/4, \sqrt{2}/4)$ ,  $\mathbf{z}_1 = (3/4, -1/4, -\sqrt{2}/4)$ . The constraint relations are then  $P_{x,y} = \{(0, 1), (1, 0), (1, 1)\} = P_{y,x}$ ,  $P_{x,z} = \{(0, 0), (0, 1), (1, 1)\} = P_{z,x}^{-1}$ ,  $P_{y,z} = \{(0, 0), (0, 1), (1, 0), (1, 1)\} = P_{z,y}$ . The pair  $(0, 0) \in P_{y,z}$  is not in the composition of the relations  $P_{y,x}$  and  $P_{x,z}$  since there is no  $a \in \{0, 1\}$  such that  $(0, a) \in P_{y,x}$  and  $(a, 0) \in P_{x,z}$ .

**5.2.4. SDPOpt( $\mathcal{I}$ ) = 1 implies solution.** Finally, we note that if  $\mathcal{I}$  is an instance of the CSP with  $\text{SDPOpt}(\mathcal{I}) = 1$  and we define  $\mathcal{J}$  using vectors with the sum (4.2) equal to 1, then a solution of  $\mathcal{J}$  is necessarily a solution to  $\mathcal{I}$ . Showing that

“SDPOpt( $\mathcal{I}$ ) = 1” implies “ $\mathcal{I}$  has a solution” was suggested as a first step to prove the Guruswami-Zhou conjecture. It indeed proved to be the right direction.

**5.3. Algebraic closure of a weak Prague instance.** The proof of correctness of the robust algorithm for bounded width CSPs obtains a solution from a certain weak Prague instance  $\mathcal{J}$ . Instance  $\mathcal{J}$  is obtained from the result of an SDP algorithm on the basic SDP relaxation of the original instance. Unfortunately the constraints in  $\mathcal{J}$  does not necessarily have bounded width so we cannot directly apply Theorem 5.5. This technical difficulty is overcome using Proposition 5.7 below. Note that the solution (given by Theorem 5.5) to the instance given by Proposition 5.7 can be outside the Prague instance  $\mathcal{J}$ .

The following lemma from [3] shows a basic property of weak Prague instances.

**LEMMA 5.6.** *Let  $\mathcal{J}$  be a weak Prague instance,  $x \in V$ ,  $A \subseteq P_x$ , and let  $p$  be a pattern from  $x$  to  $x$ . Then there exists a natural number  $l$  such that  $A + lp + l'p = A + lp$  for every integer  $l'$  and, moreover,  $A \subseteq A + lp$ .*

The set  $A + lp$  from Lemma 5.6 is denoted by  $[A]_p$ . For a singleton  $A = \{a\}$  we write  $[a]_p$ . We have  $[A]_p + l'p = [A]_p$  for every integer  $l'$  and, moreover,  $A \subseteq [A]_p$ .

**PROPOSITION 5.7.** *Let  $\mathcal{J} = (V, D, \{P_{x,y} : (x,y) \in \mathcal{S}\})$  be a weak Prague instance and let  $\mathcal{F}$  be a set of operations on  $D$ . Then  $\mathcal{J}' = (V, D, \{P'_{x,y} : (x,y) \in \mathcal{S}\})$ , where*

$$P'_{x,y} = \{(f(a_1, a_2, \dots), f(b_1, b_2, \dots)) : f \in \mathcal{F}, \\ (a_1, b_1), (a_2, b_2), \dots \in P_{x,y}\},$$

*is a weak Prague instance.*

*Proof.* It is apparent that  $\mathcal{J}'$  is 1-minimal with

$$P_x^{\mathcal{J}'} = P'_x := \{f(a_1, a_2, \dots) : f \in \mathcal{F}, a_1, a_2, \dots \in P_x\}.$$

In what follows, by  $A +' p$  we mean the addition computed in the instance  $\mathcal{J}'$  while  $A + p$  is computed in  $\mathcal{J}$ . Moreover, by  $f(A_1, \dots, A_k)$  we mean the set  $\{f(a_1, \dots, a_k) : a_1 \in A_1, a_2 \in A_2, \dots, a_k \in A_k\}$ .

Before proving (P2) and (P3) we make a simple observation.

**CLAIM 1.** *If  $f \in \mathcal{F}$  is an operation of arity  $k$ ,  $x \in V$ ,  $p$  is a pattern from  $x$ , and  $A_1, \dots, A_k \subseteq P_x$ ,  $B \subseteq P'_x$  are such that  $f(A_1, A_2, \dots, A_k) \subseteq B$ , then  $f(A_1 + p, A_2 + p, \dots, A_k + p) \subseteq B +' p$ .*

*Proof.* It is enough to prove the claim for a single step  $p = (x, y)$ . The rest follows by induction. If  $b \in f(A_1 + (x, y), \dots, A_k + (x, y))$  then there exist elements  $b_1 \in A_1 + (x, y), \dots, b_k \in A_k + (x, y)$  so that  $f(b_1, b_2, \dots, b_k) = b$ . As  $b_i \in A_i + (x, y)$  there are elements  $a_i \in A_i$  such that  $(a_i, b_i) \in P_{x,y}$  for all  $1 \leq i \leq k$ . But then  $(f(a_1, a_2, \dots, a_k), f(b_1, b_2, \dots, b_k))$  is in  $P'_{x,y}$  and  $f(a_1, a_2, \dots, a_k) \in f(A_1, A_2, \dots, A_k) \subseteq B$ , therefore  $b = f(b_1, b_2, \dots, b_k) \in B +' (x, y)$ .  $\square$

Instead of (P2) for the instance  $\mathcal{J}'$  we prove (P2\*) from Lemma 5.4. Let  $(x, y)$  be a step,  $A \subseteq P'_x$ , let  $p$  be a pattern from  $y$  to  $x$  such that  $A +' (x, y) +' p = A$ , and let  $a$  be an arbitrary element of  $A +' (x, y, x)$ . As  $A +' (x, y, x) = (A +' (x, y)) +' (y, x)$ , there exist  $b \in A +' (x, y)$  such that  $(a, b) \in P'_{x,y}$ . By definition of  $P'_{x,y}$ , we can find  $f \in \mathcal{F}$  (say, of arity  $k$ ), elements  $a_1, a_2, \dots, a_k$  in  $P_x$ , and  $b_1, \dots, b_k$  in  $P_y$  so that  $(f(a_1, a_2, \dots, a_k), f(b_1, b_2, \dots, b_k)) = (a, b)$  and  $(a_i, b_i) \in P_{x,y}$  for all  $1 \leq i \leq k$ .

We consider the sets  $[b_1]_q, [b_2]_q, \dots, [b_k]_q$  for the pattern  $q = p + (x, y)$ . We take  $l$  to be the maximum of the numbers for  $b_1, \dots, b_k$  from Lemma 5.6, so  $[b_i]_q = \{b_i\} + lq$ .

We get

$$\begin{aligned} a_i \in \{b_i\} + (y, x) &\subseteq [b_i]_q + (y, x) = \\ &= [b_i]_q + p + (x, y) + (y, x) = [b_i]_q + p, \end{aligned}$$

where the first step follows from  $(a_i, b_i) \in P_{x,y}$ , the inclusion and the first equality from Lemma 5.6, and the second equality from (P2\*) for the instance  $\mathcal{J}$  (as  $([b_i]_q + p) + (x, y) + p = [b_i]_q + p$ ). Thus  $a = f(a_1, a_2, \dots, a_k)$  is an element of

$$\begin{aligned} f([b_1]_q + p, [b_2]_q + p, \dots, [b_k]_q + p) &= \\ &= f(\{b_1\} + lq + p, \dots, \{b_k\} + lq + p) \end{aligned}$$

and this set is contained in  $(A +' (x, y)) +' lq +' p = A +' (x, y) +' l(p + (x, y)) +' p = A$  by Claim 1 applied with  $A_i = \{b_i\}$  and the pattern  $lq + p$ . We have shown that every element  $a$  of  $A +' (x, y, x)$  lies in  $A$ . The other inclusion follows from 1-minimality.

To prove (P3) let  $x \in V$ ,  $A \subseteq P'_x$  and let  $p, q$  be patterns such that  $A +' p +' q = A$ . We first show that  $A \subseteq A +' p$ . Let  $a \in P'_x$ , take  $f \in \mathcal{F}$ ,  $a_1, a_2, \dots, a_k \in P_x$  such that  $f(a_1, \dots, a_k) = a$ , and find  $l$  so that  $[a_i]_{p+q} = a_i + l(p + q)$ . From (P3) for  $\mathcal{J}$  and Lemma 5.6 it follows that  $[a_i]_{p+q} + p = [a_i]_{p+q}$ . By Claim 1,  $a \in f([a_1]_{p+q}, [a_2]_{p+q}, \dots, [a_k]_{p+q}) = f([a_1]_{p+q} + p, [a_2]_{p+q} + p, \dots, [a_k]_{p+q} + p) \subseteq A +' l(p + q) +' p = A +' p$ . The same argument used for  $A +' p$  instead of  $A$  and the patterns  $q + p, q$  instead of  $p + q, p$  proves  $A +' p \subseteq A +' p +' q = A$ .  $\square$

**6. Robust algorithm for bounded width CSPs.** The final, and most technical, version of our main result follows. Theorem 3.9 is a consequence of the following fact:

**THEOREM 6.1.** *Let  $\Gamma$  be a core constraint language over  $D$  containing at most binary relations. If  $\text{CSP}(\Gamma)$  has bounded width, then there exists a randomized algorithm which given an instance  $\mathcal{I}$  of  $\text{CSP}(\Gamma)$  and an output of the basic SDP relaxation with value at least  $1 - 1/n^{4n}$  (where  $n$  is a natural number) produces an assignment with value at least  $1 - K/n$ , where  $K$  is a constant depending on  $|D|$ . The running time is polynomial in  $m$  (the number of constraints) and  $n^n$ .*

**6.1. Proof of Theorem 3.9 using Theorem 6.1.** To prove Theorem 3.9 we start with  $\Gamma$  which is the singleton expansion of a constraint language of bounded width. By Proposition 4.1 we can assume that  $\Gamma$  contains only at most binary relations.

Let  $\mathcal{I}$  be an instance of  $\text{CSP}(\Gamma)$  with  $m$  constraints and let  $1 - \varepsilon = \text{Opt}(\mathcal{I})$  where  $\varepsilon$  is sufficiently small and  $m$  sufficiently large. We need to show how to effectively find an assignment satisfying, in expectation, the promised fraction of constraints.

We first check whether  $\mathcal{I}$  has a solution. This can be done in polynomial time since  $\text{CSP}(\Gamma)$  has bounded width. If a solution exists we can find it in polynomial time by Theorem 3.4.

In the other case we know that  $\varepsilon \geq 1/m$ . We run the SDP relaxation with precision  $\delta = 1/m$  and obtain vectors with the sum (4.2) equal to  $v \geq \text{SDPOpt}(\mathcal{I}) - 1/m$ . Finally, we execute the algorithm provided in Theorem 6.1 with the following choice of  $n$ .

$$n = \left\lceil \frac{\log \omega}{4 \log \log \omega} \right\rceil, \quad \text{where } \omega = \min \left\{ \frac{1}{1-v}, m \right\}.$$



The assumption is satisfied, because  $v \geq 1 - 1/n^{4n}$  is equivalent to  $n^{4n} \leq 1/(1 - v)$  and

$$\begin{aligned} n^{4n} &= 2^{4n \log n} \leq 2^{4 \frac{\log \omega}{4 \log \log \omega} \log \frac{\log \omega}{4 \log \log \omega}} < \\ &< 2^{\frac{\log \omega}{\log \log \omega} \log \log \omega} = \omega \leq 1/(1 - v). \end{aligned}$$

The algorithm runs in time polynomial in  $m$  as  $n^n < n^{4n} \leq \omega \leq m$ . To estimate the fraction of satisfied constraints, observe that  $v \geq \text{Opt}(\mathcal{I}) - 1/m = 1 - \varepsilon - 1/m \geq 1 - 2\varepsilon$ , so  $1/(1 - v) \geq 1/(2\varepsilon)$ , and also  $m \geq 1/\varepsilon$ , therefore  $\omega \geq 1/(2\varepsilon)$ . The fraction of satisfied constraints is, in expectation, at least  $1 - K/n$  and

$$\begin{aligned} \frac{n}{K} &\geq \frac{1}{K} \left( \frac{\log \omega}{4 \log \log \omega} - 1 \right) \geq \\ &\geq K_3 \frac{\log(1/(2\varepsilon)) \geq K_4 \frac{\log(1/\varepsilon)}{\log \log(1/\varepsilon)},}{R : \text{strangelinebreaksinthesecondandthirddisplayedlines.}} \end{aligned}$$

where  $K_3, K_4$  are suitable constants. Therefore the fraction of satisfied constraints is at least

$$1 - O\left(\frac{\log \log(1/\varepsilon)}{\log(1/\varepsilon)}\right).$$

**6.2. Proof of Theorem 6.1.** Let  $\mathcal{I} = (V, D, \mathcal{C})$  be an instance of  $\text{CSP}(\Gamma)$  with  $m$  constraints and let  $\mathbf{x}_a, x \in V, a \in D$  be vectors satisfying (SDP1), (SDP2), (SDP3) such that the sum (4.2) is at least  $1 - 1/n^{4n}$ . Without loss of generality we assume that  $n > |D|$ .

Let us first briefly sketch the idea of the algorithm. The aim is to define an instance  $\mathcal{J}$  in a similar way as in the previous section ( $\mathcal{J}$  is defined after Claim 2), but instead of all pairs with nonzero weight we only include pairs of weight greater than a threshold (chosen in Step 1). This guarantees that every solution to  $\mathcal{J}$  satisfies all the constraints of  $\mathcal{I}$  which do not have large weight on pairs outside the constraint relation (the bad constraints are removed in Step 3). The instance  $\mathcal{J}$  (more precisely, its algebraic closure) has a solution by Theorem 5.5 as soon as we ensure that it is a weak Prague instance. Property (P1) is dealt with in a similar way as in [23]: We keep only constraints with a gap – all pairs have either smaller weight than the threshold, or significantly larger (Step 2). This also gives a property similar to the one in the motivating discussion in the previous section: The vector  $\mathbf{y}_{A+(x,y)}$  is either significantly longer than  $\mathbf{x}_A$  or these vectors are almost the same. However, large amount of small differences can add up, so we need to continue taming the instance. In Steps 4 and 5 we divide the unit ball into layers and remove some constraints so that almost the same vectors of the form  $\mathbf{x}_A, \mathbf{y}_{A+(x,y)}$  never lie in different layers. This already guarantees property (P2). For property (P3) we use “cutting by hyperplanes” idea from [14]. We choose sufficiently many hyperplanes so that every pair  $\mathbf{x}_A, \mathbf{x}_B$  of different vectors in the same layer is cut (the bad variables are removed in Step 7) and we do not allow almost the same vectors for different variables to cross the hyperplane (Step 8).

The description of the algorithm follows.

1. Choose  $r \in \{1, 2, \dots, n - 1\}$  uniformly at random.
2. Remove from  $\mathcal{C}$  all the unary constraints  $(x, R)$  such that  $\|\mathbf{x}_a\|^2 \in [n^{-4r-4}, n^{-4r})$  for some  $a \in D$  and all the binary constraints  $((x, y), R)$  such that  $\mathbf{x}_a \mathbf{y}_b \in [n^{-4r-4}, n^{-4r})$  for some  $a, b \in D$ .

3. Remove from  $\mathcal{C}$  all the unary constraints  $(x, R)$  such that  $\|\mathbf{x}_a\|^2 \geq n^{-4r}$  for some  $a \notin R$  and all the binary constraints  $((x, y), R)$  such that  $\mathbf{x}_a \mathbf{y}_b \geq n^{-4r}$  for some  $(a, b) \notin R$ .

Let

$$u_1 = 2|D|^2 n^{-4r-4} \quad \text{and} \quad u_2 = n^{-4r} - u_1.$$

For two real numbers  $\gamma, \psi \neq 0$  we denote by  $\gamma \div \psi$  the greatest integer  $i$  such that  $\gamma - i\psi > 0$  and this difference is denoted by  $\gamma \bmod \psi$ .

4. Choose  $s \in [0, u_2]$  uniformly at random.

5. Remove from  $\mathcal{C}$  all the binary constraints  $((x, y), R)$  such that  $\|\mathbf{x}_A\|^2 - \|\mathbf{y}_B\|^2 \leq u_1$  and  $(\|\mathbf{x}_A\|^2 - s) \div u_2 \neq (\|\mathbf{y}_B\|^2 - s) \div u_2$  for some  $A, B \subseteq D$ .

The remaining part of the algorithm uses the following definitions. For all  $x \in V$  let

$$P_x = \{a \in D : \|\mathbf{x}_a\|^2 \geq n^{-4r}\}.$$

For a vector  $\mathbf{w}$  we put

$$h(\mathbf{w}) = (\|\mathbf{w}\|^2 - s) \div u_2$$

and

$$t(\mathbf{w}) = \left\lceil \pi(\log n) n^{2r} \min\{\sqrt{(h(\mathbf{w}) + 2)u_2}, 1\} \right\rceil.$$

We say that  $\mathbf{w}_1$  and  $\mathbf{w}_2$  are *almost the same* if  $h(\mathbf{w}_1) = h(\mathbf{w}_2)$  and  $\|\mathbf{w}_1 - \mathbf{w}_2\|^2 \leq u_1$ .

6. Choose unit vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{\lceil \pi(\log n) n^{2r} \rceil}$  independently and uniformly at random.
7. We say that a variable  $x \in V$  is *uncut* if there exists  $A, B \subseteq P_x$ ,  $A \neq B$  such that  $h(\mathbf{x}_A) = h(\mathbf{x}_B)$  and  $\text{sgn } \mathbf{x}_A \mathbf{q}_i = \text{sgn } \mathbf{x}_B \mathbf{q}_i$  for every  $1 \leq i \leq t(\mathbf{x}_A)$  (in words, no hyperplane determined by the first  $t(\mathbf{x}_A) = t(\mathbf{x}_B)$  vectors  $\mathbf{q}_i$  cuts the vectors  $\mathbf{x}_A, \mathbf{x}_B$ ). Remove from  $\mathcal{C}$  all the constraints whose scope contains an uncut variable.
8. Remove from  $\mathcal{C}$  all the binary constraints  $((x, y), R)$  for which there exist  $A \subseteq P_x, B \subseteq P_y$  such that  $\mathbf{x}_A, \mathbf{y}_B$  are almost the same and  $\text{sgn } \mathbf{x}_A \mathbf{q}_i \neq \text{sgn } \mathbf{y}_B \mathbf{q}_i$  for some  $1 \leq i \leq t(\mathbf{x}_A)$ .
9. Use the remaining constraints to construct a weak Prague instance, close it under polymorphisms (comp. Proposition 5.7) and compute a solution.

CLAIM 2. *Expected fraction of constraints removed in steps 2, 3, 5, 7 and 8 is at most  $K/n$  for some constant  $K$ .*

**Remark.** The constant  $K$  depends exponentially on the size of the domain  $|D|$ .

*Proof.*

**Step 2.** For each binary constraint there are  $|D|^2$  choices for  $a, b \in D$  and therefore at most  $|D|^2$  bad choices for  $r$ . For a unary constraint the number of bad choices is at most  $|D|$ . Thus the probability that a given constraint will be removed is at most  $|D|^2/(n-1)$  and it follows that the expected fraction of removed constraints is at most  $|D|^2/(n-1)$ .

**Step 3.** The contribution of every removed constraint to the sum (4.2) is at most  $1 - n^{-4r} \leq 1 - n^{-4n+4}$ . If more than  $\gamma$ -fraction of the constraints is removed than the sum is at most  $1/m((1-\gamma)m + \gamma m(1 - n^{-4n+4})) = 1 - \gamma n^{-4n+4}$ . Since (4.2)  $\geq 1 - 1/n^{4n}$ , we have  $\gamma \leq 1/n^4$ .

**Step 5.** For every constraint  $((x, y), R)$  and every  $A, B \subseteq D$  such that  $||\mathbf{x}_A||^2 - ||\mathbf{y}_B||^2 \leq u_1$ ,  $||\mathbf{x}_A|| \leq ||\mathbf{y}_B||$ , the inequality  $(||\mathbf{x}_A||^2 - s) \div u_2 < (||\mathbf{y}_B||^2 - s) \div u_2$  can be satisfied only if  $(||\mathbf{y}_B||^2 - s) \bmod u_2 < u_1$ . The bad choices for  $s$  thus cover at most  $(u_1/u_2)$ -fraction of the interval  $[0, u_2]$ . As  $u_1/u_2 < K_1/n^4$  (for a suitable constant  $K_1$  depending on  $|D|$ ), the probability of a bad choice is at most  $K_1/n^4$ . There are  $4^{|D|}$  pairs of subsets  $A, B \subseteq D$ , therefore the probability that the constraint is removed is less than  $K_1 4^{|D|}/n^4$  and so is the expected fraction of removed constraints.

Before analyzing Steps 7 and 8 let us observe that, for any vector  $\mathbf{w}$  such that  $1 \geq ||\mathbf{w}|| \geq n^{-4r}$ ,

$$\pi(\log n)n^{2r} ||\mathbf{w}|| \leq t(\mathbf{w}) \leq 2\pi(\log n)n^{2r} ||\mathbf{w}|| + 1.$$

The first inequality follows from

$$\begin{aligned} \sqrt{(h(\mathbf{w}) + 2)u_2} &= \sqrt{u_2((||\mathbf{w}||^2 + 2u_2 - s) \div u_2)} \geq \\ &\geq \sqrt{u_2 \frac{||\mathbf{w}||^2 + u_2 - s}{u_2}} \geq ||\mathbf{w}|| \end{aligned}$$

and the second inequality follows from

$$\begin{aligned} \sqrt{(h(\mathbf{w}) + 2)u_2} &\leq \sqrt{u_2 \frac{(||\mathbf{w}||^2 + 2u_2 - s)}{u_2}} \leq \\ &\leq \sqrt{||\mathbf{w}||^2 + 2u_2} \leq \sqrt{||\mathbf{w}||^2 + 2||\mathbf{w}||^2} < 2||\mathbf{w}||. \end{aligned}$$

**Step 7.** Consider two different subsets  $A, B$  of  $P_x$  such that  $h(\mathbf{x}_A) = h(\mathbf{x}_B)$ . Suppose that  $A \setminus B \neq \emptyset$ , the other case is symmetric. Let  $\theta$  be the angle between  $\mathbf{x}_A$  and  $\mathbf{x}_B$ . As  $\mathbf{x}_A - \mathbf{x}_{A \cap B} (= \mathbf{x}_{A \setminus B})$ ,  $\mathbf{x}_B - \mathbf{x}_{A \cap B}$  and  $\mathbf{x}_{A \cap B}$  are pairwise orthogonal, the angle  $\theta$  is greater than or equal to the angle  $\theta_A$  between  $\mathbf{x}_A$  and  $\mathbf{x}_{A \cap B}$ . (Given three pairwise orthogonal vectors  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ , the angle between  $\mathbf{v}_1 + \mathbf{v}_2$  and  $\mathbf{v}_1 + \mathbf{v}_3$  is always greater than or equal to the angle between  $\mathbf{v}_1 + \mathbf{v}_2$  and  $\mathbf{v}_1$ . This is a straightforward calculation using, for instance, dot products. In our situation  $\mathbf{v}_1 = \mathbf{x}_{A \cap B}$ ,  $\mathbf{v}_2 = \mathbf{x}_{A \setminus B}$  and  $\mathbf{v}_3 = \mathbf{x}_{B \setminus A}$ .) We have  $\sin \theta_A = ||\mathbf{x}_{A \setminus B}|| / ||\mathbf{x}_A||$ . Since  $A \subseteq P_x$ , we get  $||\mathbf{x}_{A \setminus B}|| \geq \sqrt{n^{-4r}} = n^{-2r}$  and then  $\sin \theta_A = ||\mathbf{x}_{A \setminus B}|| / ||\mathbf{x}_A|| \geq n^{-2r} / ||\mathbf{x}_A||$ , so  $\theta \geq \theta_A \geq n^{-2r} / ||\mathbf{x}_A||$ .

The probability that  $\mathbf{q}_i$  does not cut  $\mathbf{x}_A$  and  $\mathbf{x}_B$  is thus at most  $1 - n^{-2r} / \pi ||\mathbf{x}_A||$  and the probability that none of the vectors  $\mathbf{q}_1, \dots, \mathbf{q}_{t(\mathbf{x}_A)}$  cut them is at most

$$\begin{aligned} \left(1 - \frac{n^{-2r}}{\pi ||\mathbf{x}_A||}\right)^{t(\mathbf{x}_A)} &\leq \left[\left(1 - \frac{1}{\pi n^{2r} ||\mathbf{x}_A||}\right)^{\pi n^{2r} ||\mathbf{x}_A||}\right]^{\log n} \leq \\ &\leq \left(\frac{1}{2}\right)^{\log n} = \frac{1}{n}. \end{aligned}$$

The first inequality uses that  $t(\mathbf{x}_A) \geq \pi(\log n)n^{2r} ||\mathbf{x}_A||$  which we observed above. In the second inequality we have used that  $(1 - 1/\eta)^\eta \leq 1/2$  whenever  $\eta \geq 2$ .

For a single variable there are at most  $4^{|D|}$  choices for  $A, B \subseteq P_x$ , therefore the probability that  $x$  is uncut is at most  $4^{|D|}/n$ . The scope of every constraint contains at most 2 variables, hence the probability that a constraint is removed is at most  $2 \cdot 4^{|D|}/n$  and the expected fraction of the constraints removed in this step has the same upper bound.

**Step 8.** Assume that  $((x, y), R)$  is a binary constraint and  $A \subseteq P_x, B \subseteq P_y$  are such that  $\mathbf{x}_A$  and  $\mathbf{y}_B$  are almost the same. Let  $\theta$  be the angle between  $\mathbf{x}_A$  and  $\mathbf{y}_B$  and  $\theta_A$  be the angle between  $\mathbf{y}_B$  and  $\mathbf{y}_B - \mathbf{x}_A$ . By the law of sines we have  $\|\mathbf{x}_A\|/(\sin \theta_A) = \|\mathbf{y}_B - \mathbf{x}_A\|/(\sin \theta)$ , and

$$\theta \leq 2 \sin \theta = \frac{2 \|\mathbf{y}_B - \mathbf{x}_A\|}{\|\mathbf{x}_A\|} \sin(\theta_A) \leq \frac{2 \|\mathbf{y}_B - \mathbf{x}_A\|}{\|\mathbf{x}_A\|} \leq \frac{2\sqrt{u_1}}{\|\mathbf{x}_A\|},$$

where the first inequality follows from  $\theta \leq \pi/2$  (as the dot product of  $\mathbf{x}_A$  and  $\mathbf{y}_B$  is a sum of nonnegative numbers). Therefore, the probability that vectors  $\mathbf{x}_A$  and  $\mathbf{y}_B$  are cut by some of the vectors  $\mathbf{q}_i$ ,  $1 \leq i \leq t(\mathbf{x}_A)$  is at most

$$\begin{aligned} t(\mathbf{x}_A) \frac{2\sqrt{u_1}}{\|\mathbf{x}_A\|} &\leq (2\pi(\log n)n^{2r} \|\mathbf{x}_A\| + 1) \frac{2\sqrt{2|D|^2 n^{-4r-4}}}{\|\mathbf{x}_A\|} \leq \\ &\leq K_2(\log n)n^{-2} \leq \frac{K_2}{n}, \end{aligned}$$

where  $K_2$  is a constant. There are at most  $4^{|D|}$  choices for  $A, B$ , so the probability that our constraint will be removed is less than  $K_2 4^{|D|}/n$ .  $\square$

Now we define the instance  $\mathcal{J}$  and proceed to show that  $\mathcal{J}$  is a weak Prague instance. Let  $\mathcal{S}$  denote the set of pairs which are the scope of some binary constraint of  $\mathcal{I}$  after Step 8, let  $V_0$  be the set of variables which are within the scope of some constraint after Step 8, and let  $\mathcal{S}^{-1} = \{(x, y) : (y, x) \in \mathcal{S}\}$ . We put

$$\begin{aligned} \mathcal{J} &= (V_0, D, \{((x, y), P_{x,y}^{\mathcal{J}}) : (x, y) \in \mathcal{S} \cup \mathcal{S}^{-1}\}), \\ P_{x,y}^{\mathcal{J}} &= \{(a, b) : \mathbf{x}_a \mathbf{y}_b \geq n^{-4r}\}. \end{aligned}$$

CLAIM 3. *The instance  $\mathcal{J}$  is 1-minimal and  $P_x^{\mathcal{J}} = P_x$ .*

*Proof.* Let  $(x, y) \in \mathcal{S}$  and take an arbitrary constraint  $((x, y), R)$  which remained in  $\mathcal{C}$ .

First we prove that  $P_{x,y} \subseteq P_x \times P_y$  for every  $a, b \in D$ . Indeed, if  $(a, b) \in P_{x,y}$  then  $\mathbf{x}_a \mathbf{y}_b \geq n^{-4r}$ , therefore  $\|\mathbf{x}_a\|^2 = \mathbf{x}_a \mathbf{x}_D = \mathbf{x}_a \mathbf{y}_D \geq n^{-4r}$ , so  $a \in P_x$ . Similarly,  $b \in P_y$ .

On the other hand, if  $a \in P_x$  then  $n^{-4r} \leq \|\mathbf{x}_a\|^2 = \mathbf{x}_a \mathbf{y}_D$ , thus there exist  $b \in D$  such that  $\mathbf{x}_a \mathbf{y}_b \geq n^{-4r}/|D| \geq n^{-4r-4}$  (we have used  $n^4 \geq |D|$ ). But then  $\mathbf{x}_a \mathbf{y}_b \geq n^{-4r}$ , otherwise the constraint  $((x, y), R)$  would be removed in Step 2. This implies that  $(a, b) \in P_{x,y}$ . We have shown that the projection of  $P_{x,y}$  to the first coordinate contains  $P_x$ .  $\square$

For verification of properties (P2) and (P3) the following observation will be useful.

CLAIM 4. *Let  $(x, y) \in \mathcal{S} \cup \mathcal{S}^{-1}$ ,  $A \subseteq P_x$ ,  $B = A + (x, y)$ . If  $A = B + (y, x)$ , then the vectors  $\mathbf{x}_A$  and  $\mathbf{y}_B$  are almost the same. In the other case, i.e. if  $A \subsetneq B + (y, x)$ , then  $h(\mathbf{y}_B) > h(\mathbf{x}_A)$ .*

*Proof.* The number  $\|\mathbf{y}_B - \mathbf{x}_A\|^2$  is equal to

$$\mathbf{y}_B \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_B + \mathbf{x}_A \mathbf{x}_A =$$

$$= \mathbf{x}_D \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_B + \mathbf{x}_A \mathbf{y}_D = \mathbf{x}_{D \setminus A} \mathbf{y}_B + \mathbf{x}_A \mathbf{y}_{D \setminus B}.$$

No pair  $(a, b)$ , with  $a \in A$  and  $b \in D \setminus B$ , is in  $P_{x,y}^{\mathcal{J}}$  so the dot product  $\mathbf{x}_a \mathbf{y}_b$  is smaller than  $n^{-4r}$ . Then in fact  $\mathbf{x}_a \mathbf{y}_b < n^{-4r-4}$  otherwise all the constraints with scope  $(x, y)$  would be removed in Step 2. It follows that the second summand is always at most  $|D|^2 n^{-4r-4}$  and the first summand has the same upper bound in the case  $B + (y, x) = A$ .

Moreover,  $\|\mathbf{y}_B\|^2 - \|\mathbf{x}_A\|^2$  is equal to

$$\begin{aligned} \mathbf{y}_B \mathbf{y}_B - \mathbf{x}_A \mathbf{x}_A &= \mathbf{x}_D \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_D = \\ &= \mathbf{x}_D \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_{D \setminus B} = \mathbf{x}_{D \setminus A} \mathbf{y}_B - \mathbf{x}_A \mathbf{y}_{D \setminus B}. \end{aligned}$$

If  $B + (y, x) = A$  then we have a difference of two nonnegative numbers each less than or equal  $|D|^2 n^{-4r-4}$ , therefore the absolute value of this expression is at most  $u_1$ . But then  $h(\mathbf{x}_A) = h(\mathbf{y}_B)$ , otherwise all constraint with scope  $(x, y)$  or  $(y, x)$  would be removed in Step 5. Using the previous paragraph, it follows that  $\mathbf{x}_A$  and  $\mathbf{y}_B$  are almost the same.

If  $B + (y, x)$  properly contains  $A$  then the first summand  $\mathbf{x}_{D \setminus A} \mathbf{y}_B$  is greater than or equal to  $n^{-4r}$ , so the whole expression is at least  $n^{-4r} - |D|^2 n^{-4r-4} > u_2$  and thus  $h(\mathbf{y}_B) > h(\mathbf{x}_A)$ .  $\square$

CLAIM 5.  $\mathcal{J}$  is a weak Prague instance.

*Proof.* **(P2).** Let  $A \subseteq P_x$  and let  $p = (x_1, \dots, x_i)$  be a pattern in  $\mathcal{J}$  from  $x$  to  $x$  (i.e.  $x_1 = x_i = x$ ). By the previous claim  $h(\mathbf{x}_A) = h((\mathbf{x}_i)_{A+(x_1, \dots, x_i)}) \geq h((\mathbf{x}_{i-1})_{A+(x_1, \dots, x_{i-1})}) \geq \dots \geq h((\mathbf{x}_2)_{A+(x_1, x_2)}) \geq h(\mathbf{x}_A)$ . It follows that all these inequalities must in fact be equalities and, by applying the claim again, we get that the vectors  $(\mathbf{x}_j)_{A+(x_1, x_2, \dots, x_j)}$  and  $(\mathbf{x}_{j+1})_{A+(x_1, x_2, \dots, x_{j+1})}$  are almost the same and, moreover,  $A + (x_1, x_2, \dots, x_{j+1}) + (x_{j+1}, x_j) = A + (x_1, x_2, \dots, x_j)$  for every  $1 \leq j < i$ . Therefore  $A + p - p = A$  as required.

**(P3).** Let  $A \subseteq P_x$ , let  $p_1 = (x_1, \dots, x_i), p_2$  be two patterns from  $x$  to  $x$  such that  $A + p_1 + p_2 = A$  and let  $B = A + p_1$ . For contradiction assume  $A \neq B$ . The same argument as above proves that the vectors  $(\mathbf{x}_j)_{A+(x_1, x_2, \dots, x_j)}$  and  $(\mathbf{x}_{j+1})_{A+(x_1, x_2, \dots, x_{j+1})}$  are almost the same for every  $1 \leq j < i$ , and then  $h(\mathbf{x}_A) = h(\mathbf{x}_B)$ . There exists  $k \leq t(\mathbf{x}_A)$  such that  $\text{sgn } \mathbf{x}_A \mathbf{q}_k \neq \text{sgn } \mathbf{x}_B \mathbf{q}_k$ , otherwise  $x$  is uncut and all constraints whose scope contains  $x$  would be removed in Step 7. But this leads to a contradiction, since  $\text{sgn } (\mathbf{x}_j)_{A+(x_1, \dots, x_j)} \mathbf{q}_k = \text{sgn } (\mathbf{x}_{j+1})_{A+(x_1, \dots, x_{j+1})} \mathbf{q}_k$  for all  $1 \leq j < i$ , otherwise the constraints with scope  $(x_j, x_{j+1})$  would be removed in Step 8.  $\square$

Observe that for every unary constraint  $(x, R)$  we have  $P_x \subseteq R$  (from Step 3) and for every binary constraint  $((x, y), R)$  we have  $P_{x,y} \subseteq R$ . Since we have removed at most  $(K/n)$ -fraction of the constraints from  $\mathcal{C}$ , a potential solution to  $\mathcal{J}$  is an assignment for the original instance  $\mathcal{I}$  of value at least  $1 - K/n$ . Also, the instance  $\mathcal{J}$  is nontrivial because, for each  $x \in V$ , there exists at least one  $a \in D$  with  $\|\mathbf{x}_a\|^2 > 1/n^4$  (recall that we assume  $n > |D|$ ).

The only problem is that the CSP over the constraint language of  $\mathcal{J}$  (consisting of  $P_{x,y}^{\mathcal{J}}$ 's) does not necessarily have bounded width. This is why we form the algebraic closure  $\mathcal{J}'$  of  $\mathcal{J}$ :

$$\begin{aligned} \mathcal{J}' &= (V_0, D, \{((x, y), P_{x,y}^{\mathcal{J}'}): (x, y) \in \mathcal{S} \cup \mathcal{S}^{-1}\}), \\ P_{x,y}^{\mathcal{J}'} &= \{(f(a_1, a_2, \dots), f(b_1, b_2, \dots)): f \in \text{Pol}(\Gamma), \\ &\quad (a_1, b_1), (a_2, b_2), \dots \in P_{x,y}^{\mathcal{J}}\} \end{aligned}$$

The new instance still has the property that  $P_x^{\mathcal{J}'}$  (which is equal to  $\{f(a_1, a_2, \dots) : f \in \text{Pol}(\Gamma), a_1, a_2, \dots \in P_x\}$ ) is a subset of  $R$  for every unary constraint  $(x, R)$ , and  $P_{x,y}^{\mathcal{J}'} \subseteq R$  for every binary constraint  $((x, y), R)$ , since the constraint relations are preserved by every polymorphism of  $\Gamma$ . Moreover, every polymorphism of  $\Gamma$  is a polymorphism of the constraint language  $\Lambda'$  of  $\mathcal{J}'$ , therefore  $\text{CSP}(\Lambda')$  has bounded width (see, for instance, Theorem 3.7; technically,  $\Lambda'$  does not need to be a core, but we can simply add to  $\Lambda'$  all the singleton unary relations).

By Proposition 5.7,  $\mathcal{J}'$  is a weak Prague instance. Therefore  $\mathcal{J}'$  (and thus  $\mathcal{I}$  after Step 8) has a solution by Theorem 5.5. Clearly steps 1 to 8 can be done in time polynomial with respect to  $m$  and  $n^n$ , the closure required by Proposition 5.7 is computed in time linear in  $m$  and then a solution to  $\mathcal{J}'$  can be found in polynomial time by Theorem 3.4. This concludes the proof.

**6.3. Derandomization.** The following theorem is a deterministic version of Theorem 6.1. The statement is almost the same except the running time is polynomial in  $2^{n^2 \log^2 n}$  instead of  $n^n$ .

**THEOREM 6.2.** *Let  $\Gamma$  be a core constraint language over  $D$  containing at most binary relations. If  $\text{CSP}(\Gamma)$  has bounded width, then there exists a deterministic algorithm which given an instance  $\mathcal{I}$  of  $\text{CSP}(\Gamma)$  and an output of the basic SDP relaxation with value at least  $1 - 1/n^{4n}$  (where  $n$  is a natural number) produces an assignment with value at least  $1 - K/n$ , where  $K$  is a constant depending on  $|D|$ . The running time is polynomial in  $m$  (the number of constraints) and  $2^{n^2 \log^2 n}$ .*

*Proof.* The algorithm is the same as in the proof of Theorem 6.1 except we need to avoid the random choices in Steps 1, 4 and 6.

The random choices in Step 1 and Step 4 can be easily avoided. In Step 1 we can try all  $(n - 1)$  possible choices for  $r$  and in Step 4 we can try all choices for  $s$  from some sufficiently dense finite set, for instance  $\{0, u_2/n^4, 2u_2/n^4, \dots\}$ . The only difference is that bad choices for  $s$  can cover a slightly bigger part of the discrete set than  $u_1/u_2$  (namely  $(u_2/n^4 + u_1)/u_2$ ) and we get a slightly worse constant  $K_1$ .

For the derandomization of Step 6 we first increase the constant in the definition of  $t(\mathbf{w})$ , say  $t(\mathbf{w}) = \lceil 4(\log n) \dots \rceil$ . Next we use Theorem 1.3. in [19] from which it follows that we can find (in polynomial time with respect to  $|Q|$ ) a set  $Q$  of unit vectors such that

$$|Q| = (|V||D|)^{1+o(1)} 2^{O(\log^2(1/\kappa))}$$

and such that, for any vectors  $\mathbf{v}, \mathbf{w}$  with angle  $\theta$  between them, the probability that a randomly chosen vector from  $Q$  cuts  $\mathbf{v}$  and  $\mathbf{w}$  differs from  $\theta/\pi$  by at most  $\kappa$ . We choose  $\kappa = 1/n^{2n} = 1/2^{2n \log n}$ , therefore

$$|Q| \leq K_5 m^{K_6} (2^{n^2 \log^2 n})^{K_7}$$

where we have used  $|V| = O(m)$  which is true whenever every variable is in the scope of some constraint (we can clearly make this assumption without loss of generality).

Now if we choose  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{\lceil 4(\log n)n^{2n} \rceil}$  from  $Q$  independently uniformly at random, the estimates derived in Steps 7 and 8 remain almost unchanged: The probability that  $\mathbf{q}_i$  does not cut  $\mathbf{x}_A$  and  $\mathbf{x}_B$  in Step 7 is at most  $1 - n^{-2r}/\pi \|\mathbf{x}_A\| + \kappa \leq 1 - n^{-2r}/4 \|\mathbf{x}_A\|$  (for a sufficiently large  $n$ ), and the probability in Step 8 that vectors  $\mathbf{x}_A$  and  $\mathbf{y}_B$  are cut by  $\mathbf{q}_i$  is at most  $2\sqrt{u_1}/\|\mathbf{x}_A\| + \kappa \leq 4\sqrt{u_1}/\|\mathbf{x}_A\|$ .

Unfortunately we cannot try all possible  $\lceil 4(\log n)n^{2n} \rceil$ -tuples of vectors from  $Q$  as there are too many of them. To resolve this problem we apply the method of



conditional expectations with a pessimistic estimate. We choose the vectors one by one keeping the sum of two estimates,  $E_i^{(7)}$  and  $E_i^{(8)}$ , reasonably small, where  $E_i^{(7)}$  is an estimate of the number of uncut pairs of vectors  $\mathbf{x}_A, \mathbf{x}_B$  in the same layer (i.e.  $t(\mathbf{x}_A) = t(\mathbf{x}_B)$ ) after we already chose vectors  $\mathbf{q}_1, \dots, \mathbf{q}_i$  (important in Step 7) and  $E_i^{(8)}$  is an estimate of the number of cut pairs of almost the same vectors  $\mathbf{x}_A, \mathbf{y}_B$  (important in Step 8).

To define the pessimistic estimates, we create two lists of pairs of vectors. The list  $\mathcal{L}^{(7)}$  contains the pairs we want to cut because of Step 7: For every constraint  $C$  we include to  $\mathcal{L}^{(7)}$  the pairs  $(\mathbf{x}_A, \mathbf{x}_B)$  such that  $x$  is in the scope of  $C$ ,  $A \neq B \subseteq P_x$ , and  $h(\mathbf{x}_A) = h(\mathbf{x}_B)$ . One pair  $(\mathbf{x}_A, \mathbf{x}_B)$  can appear more than once in the list — the number of occurrences is equal to the number of constraints whose scope contains  $x$ . The other list,  $\mathcal{L}^{(8)}$ , consists of those pairs which we do not want to cut because of Step 8. For each constraint  $C = ((x, y), R)$  we add to  $\mathcal{L}^{(8)}$  the pairs  $(\mathbf{x}_A, \mathbf{y}_B)$  such that  $A, B \subseteq P_x$  and  $\mathbf{x}_A$  is almost the same as  $\mathbf{y}_B$ . We denote by  $p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)$  the upper bound derived above for the probability that a random vector from  $Q$  does not cut  $\mathbf{x}_A$  and  $\mathbf{x}_B$  and by  $p^{(8)}(\mathbf{x}_A, \mathbf{y}_B)$  the upper bound for the probability that a random vector from  $Q$  cuts  $\mathbf{x}_A$  and  $\mathbf{y}_B$ , i.e.

$$p^{(7)}(\mathbf{x}_A, \mathbf{x}_B) = 1 - \frac{n^{-2r}}{4 \|\mathbf{x}_A\|}, \quad p^{(8)}(\mathbf{x}_A, \mathbf{y}_B) = 4\sqrt{u_1} / \|\mathbf{x}_A\|.$$

Suppose we have already selected vectors  $\mathbf{q}_1, \dots, \mathbf{q}_i$ ,  $0 \leq i$ . Let us denote by  $\mathcal{L}_i^{(7)}$  the sublist of  $\mathcal{L}^{(7)}$  formed by pairs which are not cut by the vectors  $\mathbf{q}_1, \dots, \mathbf{q}_i$  and by  $z_i^{(8)}$  the number of pairs from  $\mathcal{L}^{(8)}$  cut by these vectors. If we now choose vectors  $\mathbf{q}_{i+1}, \mathbf{q}_{i+2}, \dots$  from  $Q$  independently uniformly at random we can give an upper estimate for the expected fraction of pairs from  $\mathcal{L}^{(7)}$  which will remain uncut

$$E_i^{(7)} = \frac{1}{|\mathcal{L}^{(7)}|} \sum_{(\mathbf{x}_A, \mathbf{x}_B) \in \mathcal{L}_i^{(7)}} p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)^{\max(t(\mathbf{x}_A) - i, 0)},$$

and for the expected fraction of pairs from  $\mathcal{L}^{(8)}$  which are cut

$$E_i^{(8)} = \frac{1}{|\mathcal{L}^{(8)}|} \left( z_i^{(8)} + \sum_{(\mathbf{x}_A, \mathbf{y}_B) \in \mathcal{L}^{(8)}} \max(t(\mathbf{x}_A) - i, 0) p^{(8)}(\mathbf{x}_A, \mathbf{y}_B) \right).$$

Calculations made in the proof of Theorem 6.1 show that

$$E_0^{(7)} = O\left(\frac{1}{n}\right), \quad E_0^{(8)} = O\left(\frac{1}{n}\right).$$

The estimates are becoming less pessimistic as  $i$  increases. More precisely, given vectors  $\mathbf{q}_1, \dots, \mathbf{q}_i$ , if we choose  $\mathbf{q}_{i+1}$  uniformly at random from  $Q$ , the expected value of  $E_{i+1}^{(7)} + E_{i+1}^{(8)}$  is at most  $E_i^{(7)} + E_i^{(8)}$ . Indeed, if  $(\mathbf{x}_A, \mathbf{x}_B)$  is in  $\mathcal{L}_i^{(7)}$  and  $t(\mathbf{x}_A) \leq i$  then the contribution of this pair to both sums  $E_i^{(7)}$  is one and to  $E_{i+1}^{(7)}$  is zero or one. If, on the other hand,  $t(\mathbf{x}_A) > i$  then the contribution of this pair to the sum in  $E_i^{(7)}$  is  $p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)^{t(\mathbf{x}_A) - i}$  and the contribution to the sum in  $E_{i+1}^{(7)}$  is either zero, when the pair is cut by  $\mathbf{q}_{i+1}$ , or is equal to  $p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)^{t(\mathbf{x}_A) - i - 1}$ , when the pair is not cut by  $\mathbf{q}_{i+1}$ . The latter option happens with probability at most  $p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)$ . Expected contribution of the pair to the sum in  $E_{i+1}^{(7)}$  is therefore at

most  $p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)^{t(\mathbf{x}_A)-i-1} = p^{(7)}(\mathbf{x}_A, \mathbf{x}_B)^{t(\mathbf{x}_A)-i}$  which is the same as the contribution of this pair to the sum in  $E_i^{(7)}$ . We conclude that the expected value of  $E_{i+1}^{(7)}$  is less than or equal to  $E_i^{(7)}$ . Similarly, the expected value of  $E_{i+1}^{(8)}$  is at most  $E_i^{(8)}$ . It follows that the expected value of  $E_{i+1}^{(7)} + E_{i+1}^{(8)}$  is less than or equal to  $E_i^{(7)} + E_i^{(8)}$  as claimed.

This leads to the following (deterministic) algorithm. For  $i = 0, 1, \dots, \lceil 4(\log n)n^{2n} \rceil - 1$  we select any  $\mathbf{q}_{i+1} \in Q$  such that  $E_{i+1}^{(7)} + E_{i+1}^{(8)} \leq E_i^{(7)} + E_i^{(8)}$ . It remains to observe that this choice of vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots$  gives us a sufficient upper bound on the fraction of constraints removed in Steps 7 and 8. We denote by  $w^{(7)}$  the fraction of pairs in  $\mathcal{L}^{(7)}$  which are not cut by the selected vectors and  $w^{(8)}$  the fraction of pairs in  $\mathcal{L}^{(8)}$  which are cut by these vectors. By construction,  $w^{(7)} + w^{(8)} \leq E_0^{(7)} + E_0^{(8)}$ . In Step 7, a constraint  $C$  is removed if some pair  $(\mathbf{x}_A, \mathbf{x}_B)$  from  $\mathcal{L}^{(7)}$ , where  $x$  is in the scope of  $C$ , is not cut. The number of such pairs is at most  $2 \cdot 4^{|D|}$ . It follows that the fraction of constraints removed in this step is at most  $2 \cdot 4^{|D|}w^{(7)}$ . Similarly, in Step 8 we remove at most  $4^{|D|}w^{(8)} < 2 \cdot 4^{|D|}w^{(8)}$  fraction of the constraints. Altogether, we remove at most the following fraction of constraints.

$$2 \cdot 4^{|D|}(w^{(7)} + w^{(8)}) \leq 2 \cdot 4^{|D|} \left( E_0^{(7)} + E_0^{(8)} \right) = O\left(\frac{1}{n}\right)$$

□

Finally, we prove the deterministic version of the main theorem.

**THEOREM 6.3.** *If CSP( $\Gamma$ ) has bounded width then it is robustly solvable. The algorithm returns an assignment satisfying  $(1 - O(\log \log(1/\varepsilon)/\sqrt{\log(1/\varepsilon)}))$ -fraction of the constraints given a  $(1 - \varepsilon)$ -satisfiable instance.*

*Proof.* The proof is almost the same as for Theorem 3.9 except we need to ensure that  $2^{n^2 \log^2 n}$  is polynomial in  $m$ . Therefore we need to choose a smaller value for  $n$ , say

$$n = \left\lfloor \frac{\sqrt{\log \omega}}{4 \log \log \omega} \right\rfloor.$$

The inequality  $v \geq 1 - 1/n^{4n}$  still holds since the expression on the right hand side is even smaller than in Theorem 3.9. The algorithm runs in polynomial time as

$$2^{n^2 \log^2 n} \leq 2^{\left(\frac{\sqrt{\log \omega}}{4 \log \log \omega}\right)^2 \log^2 \left(\frac{\sqrt{\log \omega}}{4 \log \log \omega}\right)} \leq 2^{\left(\frac{\sqrt{\log \omega}}{4 \log \log \omega}\right)^2 \log^2(\sqrt{\log \omega})} \leq 2^{\frac{1}{4^3} \log \omega} \leq \omega \leq m$$

and the fraction of satisfied constraint is at least  $1 - K/n$ , where

$$\begin{aligned} \frac{n}{K} &\geq \frac{1}{K} \left( \frac{\sqrt{\log \omega}}{4 \log \log \omega} - 1 \right) \geq K_3 \frac{\sqrt{\log(1/2\varepsilon)}}{\log \log(1/2\varepsilon)} \geq \\ &\geq K_4 \frac{\sqrt{\log(1/\varepsilon)}}{\log \log(1/\varepsilon)}, \end{aligned}$$

therefore the fraction of satisfied constraints is at least

$$1 - O\left(\frac{\log \log(1/\varepsilon)}{\sqrt{\log(1/\varepsilon)}}\right).$$

□

**6.4. Final remarks.** The quantitative dependence of  $g$  on  $\varepsilon$  is not very far from the (UGC-) optimal bound for **Horn- $k$ -Sat**. Is it possible to get rid of the extra  $\log \log(1/\varepsilon)$ ? The question of the optimal quantitative dependence of  $g$  on  $\varepsilon$  is discussed in more detail in [11].

The presented straightforward derandomization using a result from [19] has  $g(\varepsilon) = O(\log \log(1/\varepsilon)/\sqrt{\log(1/\varepsilon)})$ . How to improve it to match the randomized version?

## REFERENCES

- [1] L. Barto. The collapse of the bounded width hierarchy. manuscript.
- [2] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *FOCS'09: Proceedings of the 50th Symposium on Foundations of Computer Science*, pages 595–603, 2009.
- [3] L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, Jan. 2014.
- [4] V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. I, II. *Kibernetika (Kiev)*, (3):1–10; *ibid.* 1969, no. 5, 1–9, 1969.
- [5] A. Bulatov. Bounded relational width. 2009. manuscript.
- [6] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34:720–742, March 2005.
- [7] A. A. Bulatov, A. Krokhin, and B. Larose. Complexity of constraints. chapter Dualities for Constraint Satisfaction Problems, pages 93–124. Springer-Verlag, Berlin, Heidelberg, 2008.
- [8] A. A. Bulatov, A. A. Krokhin, and P. Jeavons. Constraint satisfaction problems and finite algebras. In *Automata, languages and programming (Geneva, 2000)*, volume 1853 of *Lecture Notes in Comput. Sci.*, pages 272–282. Springer, Berlin, 2000.
- [9] M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 205–214, New York, NY, USA, 2006. ACM.
- [10] M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Trans. Algorithms*, 5:32:1–32:14, July 2009.
- [11] V. Dalmau and A. Krokhin. Robust satisfiability for csps: Hardness and algorithmic results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, Nov. 2013.
- [12] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28:57–104, February 1999.
- [13] D. Geiger. Closed systems of functions and predicates. *Pacific J. Math.*, 27:95–100, 1968.
- [14] M. X. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.
- [15] V. Guruswami and Y. Zhou. Tight bounds on the approximability of almost-satisfiable Horn SAT and exact hitting set. In D. Randall, editor, *SODA*, pages 1574–1589. SIAM, 2011.
- [16] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.
- [17] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.
- [18] P. Jonsson, A. Krokhin, and F. Kuivinen. Hard constraint satisfaction problems have hard gaps at location 1. *Theor. Comput. Sci.*, 410:3856–3874, September 2009.
- [19] Z. Karnin, Y. Rabani, and A. Shpilka. Explicit dimension reduction and its applications. *SIAM Journal on Computing*, 41(1):219–249, 2012.
- [20] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.
- [21] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 767–775. ACM Press, 2002.
- [22] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37:319–357, April 2007.
- [23] G. Kun, R. O'Donnell, S. Tamaki, Y. Yoshida, and Y. Zhou. Linear programming, width-1 csps, and robust satisfaction. In S. Goldwasser, editor, *ITCS*, pages 484–495. ACM, 2012.
- [24] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410:1629–1647, April 2009.
- [25] B. Larose and L. Zádori. Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras. *Internat. J. Algebra Comput.*, 16(3):563–581, 2006.

- [26] B. Larose and L. Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.
- [27] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC'08*, pages 245–254, 2008.
- [28] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.
- [29] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.
- [30] U. Zwick. Finding almost-satisfying assignments. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 551–560, New York, NY, USA, 1998. ACM.